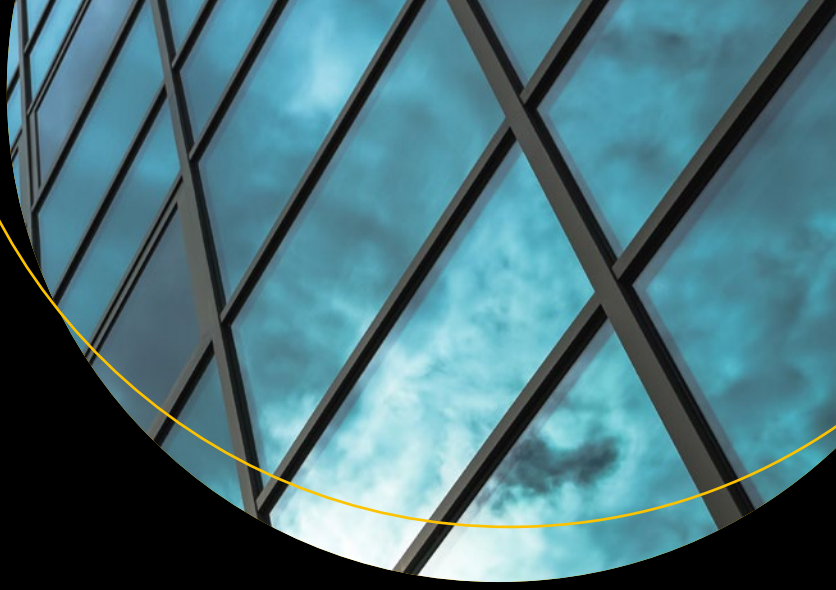


Blockchain Enabled Applications

Understand the Blockchain Ecosystem and How to Make it Work for You

Vikram Dhillon
David Metcalf
Max Hooper

Apress®

www.allitebooks.com

Blockchain Enabled Applications

Understand the Blockchain Ecosystem and How to Make it Work for You

Vikram Dhillon
David Metcalf
Max Hooper

Apress®

www.allitebooks.com

Blockchain Enabled Applications

Vikram Dhillon
Orlando, Florida, USA

David Metcalf
Orlando, Florida, USA

Max Hooper
Orlando, Florida, USA

ISBN-13 (pbk): 978-1-4842-3080-0
ISBN-13 (electronic): 978-1-4842-3081-7
https://doi.org/10.1007/978-1-4842-3081-7

Library of Congress Control Number: 2017960811

Copyright © 2017 by Vikram Dhillon, David Metcalf, and Max Hooper

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

Trademarked names, logos, and images may appear in this book. Rather than use a trademark symbol with every occurrence of a trademarked name, logo, or image we use the names, logos, and images only in an editorial fashion and to the benefit of the trademark owner, with no intention of infringement of the trademark.

The use in this publication of trade names, trademarks, service marks, and similar terms, even if they are not identified as such, is not to be taken as an expression of opinion as to whether or not they are subject to proprietary rights.

While the advice and information in this book are believed to be true and accurate at the date of publication, neither the authors nor the editors nor the publisher can accept any legal responsibility for any errors or omissions that may be made. The publisher makes no warranty, express or implied, with respect to the material contained herein.

Cover image designed by Freepik

Managing Director: Welmoed Spahr
Editorial Director: Todd Green
Acquisitions Editor: Louise Corrigan
Development Editor: James Markham
Technical Reviewer: Zeeshan Chawdhary
Coordinating Editor: Nancy Chen
Copy Editor: Teresa Horton
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit http://www.apress.com/rights-permissions.

Apress titles may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Print and eBook Bulk Sales web page at http://www.apress.com/bulk-sales.

Any source code or other supplementary material referenced by the author in this book is available to readers on GitHub via the book's product page, located at www.apress.com/9781484230800. For more detailed information, please visit http://www.apress.com/source-code.

Printed on acid-free paper

www.allitebooks.com

Vikram Dhillon would like to dedicate this work to Aaron Hillel Swartz and his legacy.

David Metcalf would like to thank Katy, Adam and Andrew for their patience during the extended hours and effort while putting the book together and colleagues and students at UCF and through the NSF I-Corps program that identified the power of Bitcoin and Blockchain technology years ago and shared their knowledge and future strategies that inspired us to pursue this area of research early. Thank you to my coauthors and our outside collaborators and contributors, and of course to God for the wisdom, ability and grit to bring this effort to life.

Max Hooper would like to thank his co-authors and colleagues at UCF/METIL Lab, along with special thanks to Mindy Hooper for her help and support. Additionally, for God's inspiration, guidance, direction and wisdom, I would like to acknowledge His leadership.

www.allitebooks.com

Contents

About the Authors xi

About the Technical Reviewer xiii

Acknowledgments xv

Introduction xvii

Chapter 1: Behold the Dreamers 1

Paradigm Shift 1

Cypherpunk Community 3

Summary 5

Chapter 2: The Gold Rush: Mining Bitcoin 7

Reaching Consensus 7

Mining Hardware 12

Startup Stories 13

New Consensus 14

Summary 14

References 14

Chapter 3: Foundations of Blockchain 15

Transaction Workflow 15

Simple Payment Verification 21

Blockchain Forks 23

Summary 24

References 24

www.allitebooks.com

Chapter 4: Unpacking Ethereum 25
Overview of Ethereum 25
Accounts in Ethereum 27
State, Storage, and Gas 30
Ethereum Virtual Machine 33
Solidity Programming Language 36
World Computer 38
Blockchain-as-a-Service 41
Decentralized Applications 42
Geth and Mist 44
Summary 44
References 45
Chapter 5: Decentralized Organizations 47
Aragon Kernel 48
Identity Management 49
DAO/Company Walkthrough 50
Setting Up a DAO 50
Issuing Shares 54
Fundraising and Bylaws 63
Summary 66
References 66
Chapter 6: The DAO Hacked 67
Introduction 67
The Team 69
The DAO 70
The ICO Highlights 72
The Hack 72
The Debate 75
The Split: ETH and ETC 76

The Future 77
Summary 78
Chapter 7: Ethereum Tokens: High-Performance Computing 79
Tokens and Value Creation 79
Ethereum Computational Market 83
Golem Network 89
Application Registry 90
Transaction Framework 91
Supercomputing Organized by Network Mining 96
Buyer–Hub–Miner Interactions 101
Superglobal Operation System for Network Architecture 104
iEx.ec 106
Summary 109
References 109
Chapter 8: Blockchain in Science 111
Reproducibility Crisis 111
Clinical Trials 115
Reputation System 119
Pharmaceutical Drug Tracking 122
Prediction Markets and Augar 123
Summary 124
Chapter 9: Blockchain in Health Care 125
Payer–Providers–Patient Model 125
Workflow 127
Hot Switching 131
Waste Management: Capital One, Ark Invest, and Gem 131
Verifiable Data Audit 134
Summary 137
References 137

Chapter 10: The Hyperledger Project 139

Current Status 139

Governance 140

Fabric and Sawtooth 141

Decision Models: Do You Need a Blockchain? 144

Rapid Prototyping with Hyperledger Composer 147

Summary 149

Chapter 11: Recent Developments in Blockchain 151

EOS Blockchain 151

Delegated Proof-of-Stake 154

Parallel Execution 157

Scheduling 159

Chain Core 160

Ethereum Enterprise Alliance 175

zk-SNARKs 177

Review of Quorum 177

Ethereum Enterprise Roadmap 180

Summary 181

References 181

Chapter 12: Technological Revolutions and Financial Capital 183

State of the Blockchain Industry 184

Blockchain Solution 184

Venture Capital and ICOs 185

Initial Coin Offerings 185

Digital Currency Exchanges 189

Status of ICO Regulation 189

Pros and Cons of ICO Investments 190

Regulation Technology: RegChain 192

New Blockchain Companies and Ideas 194
Homechain and SALT 194
Ambrosus, Numerai, and SWARM 194
Democratizing Investment Opportunities 195
Summary 196
Appendix A: Building a Health Care Consortium 197
Appendix B: References 207
Index 213

About the Authors

Vikram Dhillon is a research fellow in the Institute of Simulation and Training at the University of Central Florida where he studies the integration of emerging technologies into existing infrastructure. The focus of his recent work has been on decentralized ledger technologies. He holds a Bachelor of Science degree in Molecular Biology from the University of Central Florida, where his focus was bioinformatics. Currently, he is a DO-MBA candidate at the College of Medicine, Nova Southeastern University. He is the author of several scientific papers in computational genomics and two books, the most recent one on blockchain enabled applications. He has also written in-depth articles for the Bitcoin Magazine and letters for the *New York Times.* He was previously funded by the National Science Foundation through the Innovation Corps program to study customer discovery and apply it to commercialize high-risk startup ideas. He is a member of the Linux Foundation and has been active in open source projects and initiatives for the past several years. He often speaks at local conferences and meetups about programming, design, security, and entrepreneurship. He currently lives in Fort Lauderdale, Florida, and writes a technology-focused blog at `opsbug.com`.

David Metcalf has more than 20 years of experience in the design and research of Web-based and mobile technologies converging to enable learning and health care. Dr. Metcalf is Director of the Mixed Emerging Technology Integration Lab (METIL) at UCF's Institute for Simulation and Training. The team has built mHealth solutions, simulations, games, eLearning, mobile and enterprise IT systems for Google, J&J, the Veterans Administration, U.S. military, and the UCF College of Medicine among others. Recent projects include Lake Nona's Intelligent Home prototype and SignificantTechnology, a mobile-enabled online degree and eResource kit. Dr. Metcalf encourages spin-offs from the lab as part of the innovation process and has launched Moving Knowledge and several other for-profit and nonprofit ventures as examples. In addition to research and commercial investments, he supports social entrepreneurship in education and health. Dr. Metcalf continues to bridge the gap between corporate learning and simulation techniques and nonprofit and social entrepreneurship. Simulation, mobilization, mobile patient records and medical decision support systems, visualization systems, scalability models, secure mobile data communications, gaming, innovation management, and operational excellence are his current research topics. Dr. Metcalf frequently presents at industry and research events shaping business strategy and use of technology to improve learning, health, and human performance. He is the coeditor and author of *Connected Health* (2017), *HIMSS mHealth Innovation* (2014) and the HIMSS Books bestseller *mHealth: From Smartphones to Smart Systems* (2012).

Max Hooper is the chief executive officer of Merging Traffic. He is responsible for the company's management and growth strategy, serving as the corporate liaison to the financial services industry and various capital formation groups. Prior to starting the company, he was cofounder of Equity Broadcasting Corporation (EBC), a media company that owned and operated more than 100 television stations across the United States. He was responsible for activities in the cable, satellite, investment banking, and technology industries and during his tenure it grew to become one of the top 10 largest broadcasting companies in the country. A lifelong learner, Hooper has earned five doctorate degrees: PhD, DMin, PhD, ThD, and DMin from a variety of institutions. As an avid runner, he has completed more than 100 marathons and an additional 20 ultra-marathons, which are 50- or 100-mile runs. He has completed the Grand Slam of Ultra Running. Hooper is committed to his family and is a husband, father to five children, and grandfather to seven grandsons. He is active in many organizations and serves on various boards of directors. He works globally with several ministries and nonprofit aid groups, and was honored to speak at the United Nations in New York in 2015.

About the Technical Reviewer

Zeeshan Chawdhary is an avid technologist, with 13 years of experience in the industry. Having started his career in mobile development with J2ME in 2005, he ventured into Web development in 2006, and has extensive experience in building robust and scalable Web applications.

He has led teams to build Web and mobile apps for companies like Nokia, Motorola, Mercedes, GM, American Airlines, and Marriott while serving as chief technology officer for a San Francisco–based firm. He is based in Mumbai, India, and has also dabbled with a few startups in India, leading teams in building a Houzz+Etsy model and car rental platform, in addition to authoring a few books on iOS, Windows Phone, and iBooks.

He currently works with an international team as director of development, serving clients with technologies like Magento, Wordpress, WooCommerce, Laravel, ReactJS, and .Net.

He can be reached at `imzeeshanc@gmail.com` or on Twitter at `@imzeeshan`.

Acknowledgments

The authors would like to acknowledge our editors Nancy Chen and Louise Corrigan for their help and guidance. The figures throughout this book were made with the help of Lucidchart. All figures from external sources were used with permission.

Introduction

Blockchain technology is poised to fundamentally change our online world. This is not some kind of miraculous, cure-all, money-making solution. One specific use of blockchain such as Bitcoin, but rather the fundamental shift for the offline world ushered in by the web with easy to use access to information and the ability to make digital copies of data or content in an unprecedented ease for distribution across the globe. Hence the name, the World Wide Web. That interconnectivity has suffered fundamental problems when it comes to transactions - TRUST.

The fundamental shift that blockchain technology represents is a method for moving away from an attempt to have a central trusted authority in a massively distributed network. But instead to have multiple sources of trust that must all agree, based on an algorithm that this transaction can be trusted as valid. Furthermore, most blockchain solutions offer an immutable and enduring record of a transaction as it is hard for any trusted or untrusted source to change or modify. This presents a completely new level of security, privacy, and TRUST to our online world. As you will see throughout this book, a variety of uses, protocols, and standards make up the current blockchain ecosystem.

We also strive to strike the perfect balance between being a technical reference and a how-to handbook that shows practical examples of both current and future state use cases. While not comprehensive, we do select for several high promise areas where blockchain technology is beginning to enable applications for an entirely new industry segment. We hope this book will inform you and provides a roadmap to your success leveraging blockchain technology to enable new applications for your business.

Throughout the book, you will see many examples of applications to reinforce key points. Early examples extend beyond financial transactions to cover other aspects of FinTech, RegTech (regulation), InsuranceTech, GovTech (eVoting, licensing, records and certification), HealthTech, and many others.

In order to understand these early examples, it is necessary to explore the Blockchain history; fundamentals of distributed trust; consensus; hardware, software and encryption in the early chapters. Next, you'll learn about the network transactions and simplified payments in Blockchain fundamentals. We'll compare this with the extended capabilities if Ethereum and specific characteristics like how gas works and distributed apps along with examples of Blockchain as a Service. To further extend these capabilities, two chapters are devoted to DAO/Decentralized Organizations and the details and examples in these areas. In Chapter 7, Ethereum Tokens are highlighted for value creation with various technology and business sector examples that highlight the power of Smart Contracts to allow multiple sources of value and rules to be embedded in the transactions directly. The next three chapters- 8, 9 and 10 segment examples into Blockchain in Science, Blockchain in Healthcare, and details on the structure of the Hyperledger Project, respectively. The final two chapters, 11 and 12 explore many recent developments and future trends, particularly in ICOs and the effect on financial markets and processes. Don't miss the late-breaking Appendix with a detailed interview with the Hashed Health leadership team and their insights on Blockchain in Healthcare. We hope you find the information in this book useful as well as enjoyable as you explore The fundamentals, current best practices and future potential of Blockchain Enabled Applications. We welcome your feedback at `info@metil.org`.

CHAPTER 1

Behold the Dreamers

Why should a man intentionally live his life with one kind of anxiety followed by another?

—Imbolo Mbue

Anxiety is perhaps the best way to describe the attitude that dominated the minds of investors and the general public toward financial markets by the end of 2008. The 2008 financial crisis is considered by numerous economists to have been the worst financial crisis since the Great Depression. The years leading up to the crisis saw a flood of irresponsible mortgage lending and a massive systemic failure of financial regulation and supervision. The fallout was so immense that it threatened the collapse of large financial institutions. National governments had to intercede to bail out major banks. This chapter begins with a discussion about the 2008 financial crisis, and then we discuss the aftermath, which led to an environment where a new banking system and alternative currency such as Bitcoin could thrive. Then, we dive into the technology stack that powers Bitcoin. Remarkably, the components of this stack are not entirely new, but they are strung together in an ingenious design. Finally, we end the discussion by talking about the heightened interest in blockchain, a major technical breakthrough that has the potential to revolutionize several industries.

Paradigm Shift

Revolutions often look chaotic, but this one was brewing quietly, headed by an unknown individual under the name Satoshi Nakamoto, who dreamed of changing the financial world. Any number of parties can be blamed for the financial crisis, but the common denominator was that fundamental financial and accounting instruments used to maintain integrity of the entire system became too complex to be used efficiently. Trust, the ultimate adhesive of all financial systems, began to disappear in 2008. The regulations have since changed to prevent similar circumstances from arising, but it was clear that there was a need for autoregulation of trust between counterparties and transparency into their ability to enter any type of a sales contract. A **counterparty** is essentially the other party in a financial transaction. In other words, it is the buyer matched to a seller. In financial transactions, one of the many risks involved is called **counterparty risk,** the risk that each party involved in a contract might not be able to fulfill its side of the agreement. The systemic failure referenced earlier can now be understood in terms of counterparty risk: Both parties in the transaction were accumulating massive counterparty risk, and in the end, both parties collapsed under the terms of the contract. Imagine a similar transaction scenario involving multiple parties, and now imagine that every single player in this scenario is a major bank or insurance company that further serves millions of customers. This is just what happened during the 2008 crisis.

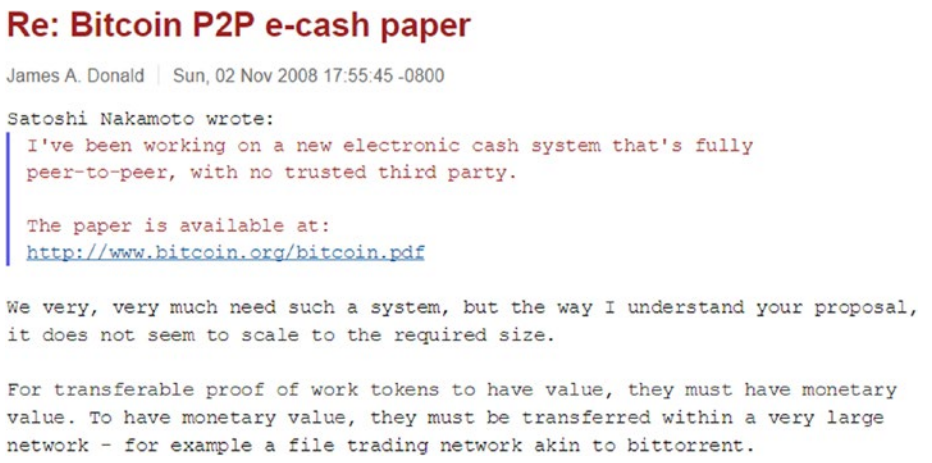
© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_1

The next issue we need to discuss is that of **double spending**. We revisit this topic again strictly in the context of Bitcoin, but let's get a basic understanding of the concept by applying it to the financial crisis. The principle behind double spending is that resources committed to one domain (e.g., one transaction) cannot also be simulataneously committed to a second disparate domain. This concept has obvious implications for digital currencies, but it can also summarize some of the problems during the 2008 crisis.

Here's how it started: Loans (in the form of mortages) were given to borrowers with poor credit histories who struggled to repay them. These high-risk mortgages were sold to financial experts at the big banks, who packaged them into low-risk public stocks by putting large numbers of them together in pools. This type of pooling would work when the risks associated with each loan (mortgage) are not correlated. The experts at big banks hypothesized that property values in different cities across the country would change independently and therefore pooling would not be risky. This proved to be a massive mistake. The pooled mortage packages were then used to purchase a type of stock called collateralized debt obligations (CDOs). The CDOs were divided into tiers and sold to investors. The tiers were ranked and rated by financial standards agencies and investors bought the safest tiers based on those ratings. Once the U.S. housing market turned, it set off a domino effect, destroying everything in the way. The CDOs turned out to be worthless, despite the ratings. The pooled mortgages collapsed in value and all the packages being sold instantly vaporized. Throughout this complex string of transactions, every sale increased the risk and incurred double spending at multiple levels. Eventually, the system equilibrated, only to find massive gaps, and collapsed under the weight. Following is a brief timeline for 2008. (This timeline was made following a presentation by Micah Winkelspech at Distributed Health, 2016).

- January 11: Bank of America buys the struggling Countrywide.
- March 16: Fed forces the sale of Bear Stearns.
- September 15: Lehman Brothers files for Chapter 11 bankruptcy.
- September 16: Fed bails out American International Group (AIG) for $85 billion.
- September 25: Washington Mutual fails.
- September 29: Financial markets crash; the Dow Jones Industrial Average fell 777.68 points and the whole system was on the brink of collapse.
- October 3: U.S. government authorizes $700 billion for bank bailouts.

The bailout had massive economic consequences, but more important, it created the type of environment that would allow for Bitcoin to flourish. In November 2008, a paper was posted on the Cryptography and Cryptography Policy Mailing List titled "Bitcoin: A Peer-to-Peer Electronic Cash System," with a single author named Satoshi Nakamoto. This paper detailed the Bitcoin protocol and along with it came the original code for early versions of Bitcoin. In some manner, this paper was a response to the economic crash that had just happened, but it would be some time before this technological revolution caught on. Some developers were concerned with this electronic cash system failing before it could ever take hold and their concern was scalability, as pointed out in Figure 1-1.


Re: Bitcoin P2P e-cash paper

James A. Donald | Sun, 02 Nov 2008 17:55:45 -0800

Satoshi Nakamoto wrote:

> I've been working on a new electronic cash system that's fully peer-to-peer, with no trusted third party.
>
> The paper is available at:
> http://www.bitcoin.org/bitcoin.pdf

We very, very much need such a system, but the way I understand your proposal, it does not seem to scale to the required size.

For transferable proof of work tokens to have value, they must have monetary value. To have monetary value, they must be transferred within a very large network - for example a file trading network akin to bittorrent.

Figure 1-1. *Initial reception of the Bitcoin protocol included concerns about scalability and realistic prospects for Bitcoin*

So who is Nakamoto? What is his background? The short and simple answer is that we don't know. In fact, it is presumptuous to assume that he is actually a "he." The name Satoshi Nakamoto was larely used as a pseudonym and he could have been a she, or even a they. Several reporters and news outlets have dedicated time and energy to digital forensics to narrow down candidates and find out the real identity of Nakamoto, but all the efforts so far have been wild-goose chases. In this case, the community is starting to realize that maybe it doesn't matter who Nakamoto is, because the nature of open source almost makes it irrelavent. Jeff Garzik, one of the most respected developers in the Bitcoin community, described it as follows: "Satoshi published an open source system for the purpose that you didn't have to know who he was, and trust who he was, or care about his knowledge." The true spirit of open source makes it so that the code speaks for itself, without any intervention from the creator or programmer.

Cypherpunk Community

Nakamoto's real genius in creating the Bitcoin protocol was solving the Byzantine generals' problem. The solution was generalized with components and ideas borrowed from the cypherpunk community. We briefly talk about three of those ideas and the components they provided for the complete Bitcoin protocol: Hashcash for proof of work, Byzantine fault tolerance for the decentralized network, and blockchain to remove the need for centralized trust or a central authority. Let's dive into each one, starting with Hashcash.

Hashcash was devised by Adam Black in the late 1990s to limit e-mail spam with the first of its kind Proof-of-Work (PoW) algorithm. The rationale behind Hashcash was to attach some computational cost to sending e-mails. Spammers have a business model that relies on sending large numbers of e-mails with very little cost associated with each message. However, if there is even a small cost for each spam e-mail sent, that cost multiplies over thousands of e-mails, making their business unprofitable. Hashcash relies on the idea of cryptographic hash functions: A type of hash function (in the case of Bitcoin, SHA1) takes an input and converts it into a string that generates a message digest, as shown in Figure 1-2. The hash functions are designed to have a property called one-way functions, which implies that a potential input can be verified very easily through the hash function to match the digest, but reproducing the input from the digest is not feasible. The only possible method of re-creating the input is by using brute force to find the appropriate input string. In practice, this is the computationally intensive element of Hashcash and also eventually Bitcoin. This principle has become the foundation behind PoW algorithms powering Bitcoin today and most cryptocurrencies. The PoW for Bitcoin is more complex and involves new components, which we talk about at length in a later chapter.

Figure 1-2. *Mechanism of a cryptographic hash function. It takes an input and consistently converts it to a string of an output digest.*

The next idea we need to discuss is the Byzantine generals' problem. It is an agreement problem among a group of generals, with each one commanding a portion of the Byzantine army, ready to attack a city. These generals need to formulate a strategy for attacking the city and communicate it to each other adequately. The key is that every general agrees on a common decision, because a tepid attack by a few generals would be worse than a coordinated attack or a coordinated retreat. The crux of the problem is that some of the generals are traitorous. They might cast a vote to deceive the other generals and ultimately lead to a suboptimal strategy. Let's take a look at an example: In a case of odd-numbered generals, say seven, three support attacking and three support retreat. The seventh general might communicate an agreement to the generals in favor of retreat, and an agreement to attack to the other generals, causing the whole arrangement to fall apart. The attacking forces fail to capture the city because no intrinsic central authority could verify the presence of trust among all seven generals.

In this scenario, Byzantine fault tolerance can be achieved if all the loyal generals can communicate effectively to reach an undisputed agreement on their strategy. If so, the misleading (faulty) vote by the traitorous general would be revealed and fail to perturb the system as a whole. For the Bitcoin protocol, Nakamoto's key innovation to enable Byzantine fault tolerance was to create a peer-to-peer network with a ledger that could record and verify a majority approval, thereby revealing any false (traitorous) transactions. This ledger provided a consistent means of communication and further allowed for removal of trust from the whole system. The ledger is also known as the blockchain, and by attaching blockchain to Bitcoin, it became the first digital currency to solve the double spending problem network-wide. In the remainder of this chapter, we present a broad overview of the broad overview of the technology, and the concept of a blockchain-enabled application.

The blockchain is primarily a recording ledger that provides all involved parties with a secure and synchronized record of transactions from start to finish. A blockchain can record hundreds of transactions very rapidly, and has several cryptographic measures intrinsic to its design for data security, consistency, and validation. Similar transactions on the blockchain are pooled together into a functional unit called a **block** and then sealed with a timestamp (a cryptographic fingerprint) that links the current block to the one preceding it. This creates an irreversible and tamper-evident string of blocks connected together by timestamps, conveniently called a blockchain. The architecture of blockchain is such that every transaction is very rapidly verified by all members of the network. Members also contain an up-to-date copy of the blockchain locally, which allows for consensus to be reached within the decentralized network. Features such as immutable record-keeping and network-wide consensus can be integrated into a stack to develop new types of applications called decentralized apps (DApps). Let's look at a prototype of a DApp in Figure 1-3, in the context of the Model-View-Controller (MVC) framework.

■ **Note** The first block of the blockchain is called the Genesis block. This block is unique in that it does not link to any blocks preceeding it. Nakmoto added a bit of historical information to this block as context for the current financial environment in the United Kingdom: "*The Times* 03/Jan/2009 Chancellor on brink of second bailout for banks. "This block not only proves that no bitcoins existed before January 3, 2009, but also gives a little insight into the mind of the creators.

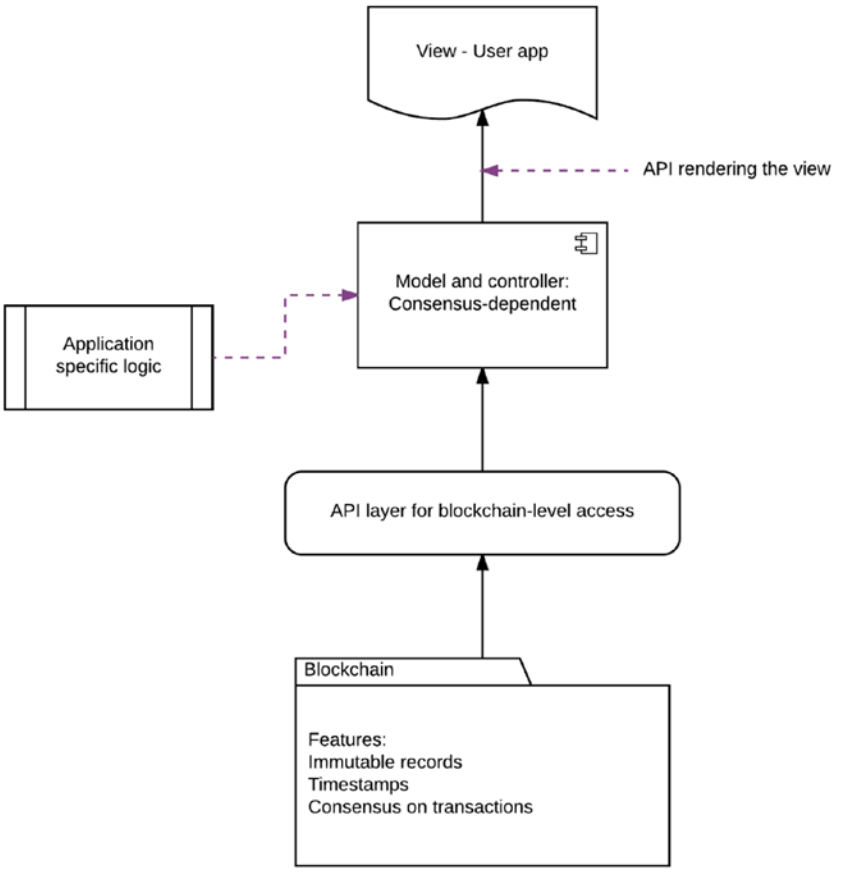


Figure 1-3. *Simple prototype of a decentralized application that interacts with the end user at the final steps*

The model and controller here rely on the blockchain for data (data integrity and security) and accordingly update the view for the end user. The secret sauce in this prototype is the application programming interface (API), which works to pull information from the blockchain and provides it to the model and controller. This API provides opportunities to extend business logic and add it to the blockchain, along with basic operations that take blocks as input and provide answers to binary questions. The blockchain could eventually have more features, such as oracles that can verify external data and timestamp it on the blockchain itself. Once a decentralized app starts dealing with large amounts of live data and sophisticated business logic, we can classify it as a blockchain-enabled application.

Summary

In this chapter, we started talking about the history of Bitcoin and the financial environment at the time it came into being. We continue our discussion of blockchain and specific features of the peer-to-peer network such as miners and more in the upcoming chapters. The references used in this chapter are available at the end of the book.

CHAPTER 2

The Gold Rush: Mining Bitcoin

During the Gold Rush, most would-be miners lost money, but people who sold them picks, shovels, tents and blue-jeans (Levi Strauss) made a nice profit.

—Peter Lynch

Mining is a foundational concept in understanding how the Bitcoin protocol operates. It refers to a decentralized review process performed on each block of the blockchain to reach consensus without the need for a central authority to provide trust. In other words, mining is the computational equivalent of peer review in a decentralized environment where neither party involved trusts the other. We continue our discussion of a hash-function explained in Chapter 1 as it refers to mining and solving PoW functions. Then, we integrate the concepts of block target values and network difficulty with mining and how mining has evolved to keep up with the increasing difficulty. This will lead us further into talking about the types of hardware mining that have recently been developed. We end the chapter with an analysis of startups that began selling dedicated hardware for mining, leading to the Bitcoin mining arms race and their eventual failure.

Reaching Consensus

Mining is central to the Bitcoin protocol and has two primary roles: adding new bitcoins to the money supply and verifying transactions. In this chapter, we look at the mechanisms behind these two processes. Essentially, mining is the appropriate solution to the double spending problem we discussed previously. To remove the need for a central authority, individuals running the Bitcoin client on their own machines (called miners) participate in the network and verify that transactions taking place between two parties are not fraudulent. Mining is actually a computationally intensive activity, but what incentives does anyone have to help mine for new Bitcoins? The key incentive for miners is getting a reward in the form of Bitcoins for their participation. Let's look at a simplified view of the mining process in Figure 2-1.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_2

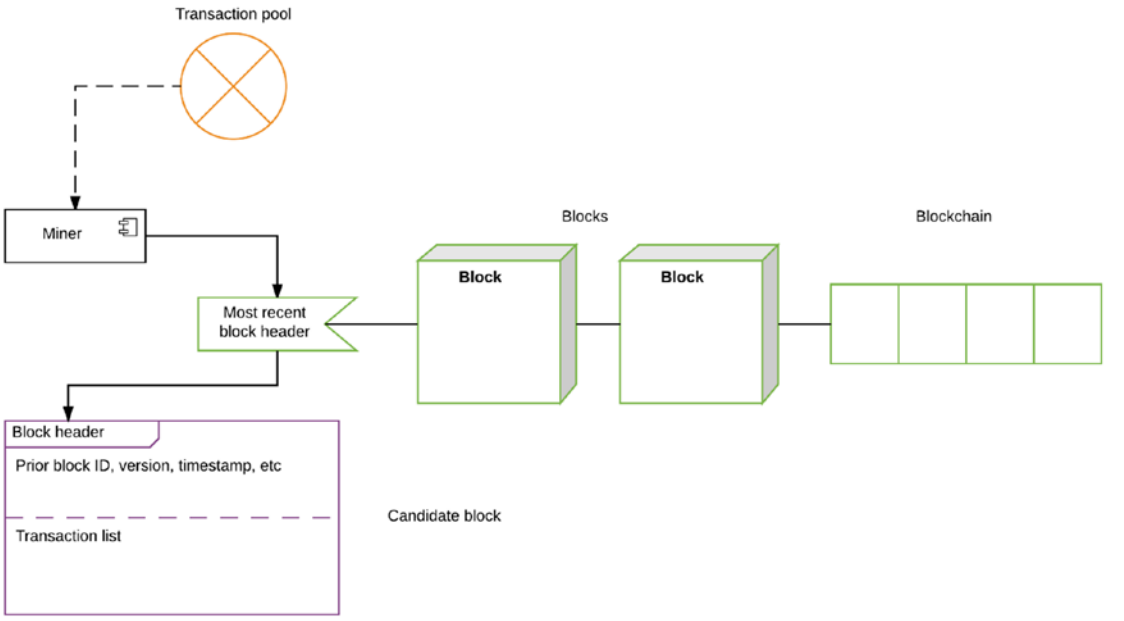


Figure 2-1. *A simplified overview of the mining process*

Unpackaged transactions that have recently occurred in the Bitcoin network remain in the transaction pool until they are picked up by a miner to be packaged into a block. A miner selects transactions from the transaction pool to package them in a block. After the block has been created, it needs a header before it can be accepted by the blockchain. Think of this as shipping a package: Once the package has been created, it needs to be stamped so that it can be shipped. A miner uses the header of the most recent block in the blockchain to construct a new header for this current block. The block header also contains other elements such as a timestamp, version of the Bitcoin client, and an ID corresponding to the previous block in the chain. The resulting block is called a candidate block, and it can now be added to the blockchain if a few other conditions are satisfied.

The process of mining is very involved and Figure 2-1 only serves to paint a broad picture regarding the participation of miners in the protocol. Next, we explore the technical aspects of the stamp (analogy referenced earlier) and the mechanism of stamping a package. Keep in mind that mining is a competitive process: Figure 2-1 describes this process for only one miner, but in reality, a very large number of miners participate in the network. The miners compete with each other to find a stamp for the package (block) they created, and the first miner to discover the stamp wins. The race between miners to find a stamp is concluded within ten minutes, and a new race begins in the next ten minutes. Once the stamp is discovered, the miner can complete the block and announce it to the network, and then it can be added to the blockchain. Let's take a look at the process behind searching for the stamp, better known as a block-header, in Figure 2-2.

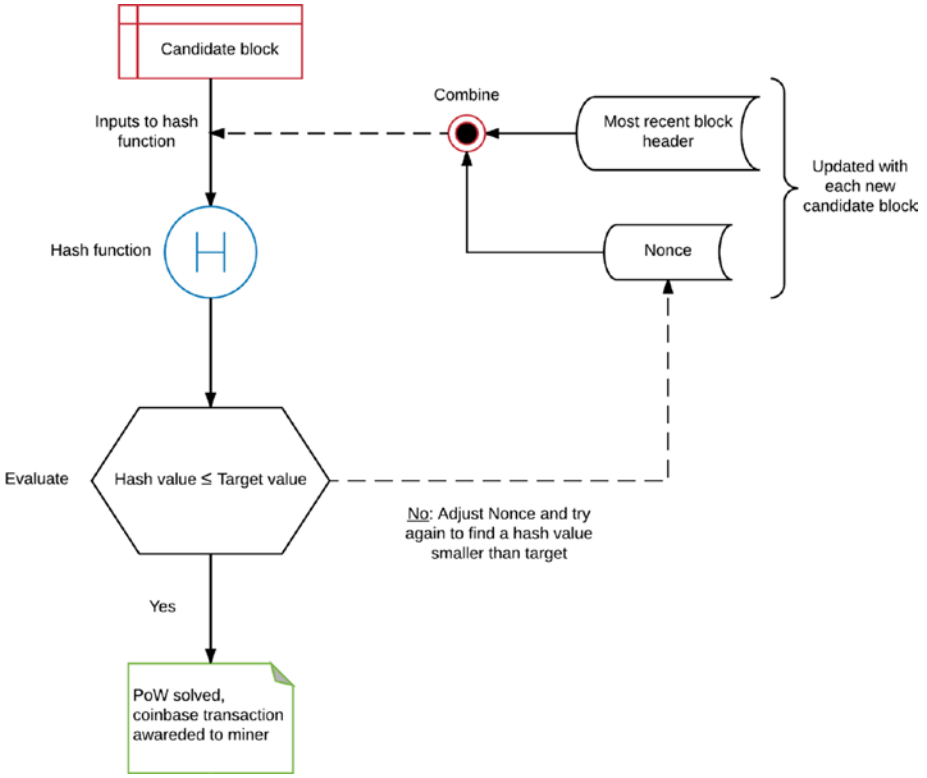


Figure 2-2. *Generating a block header by solving proof of work (PoW)*

The package created by a miner is almost a block, but it is missing a header. It's called a candidate block and it can only be added to the blockchain after the stamp, or the header, is added. The header from the most recent block in the blockchain is retrieved and combined with a 32-bit value called nonce. This combination is directed to the hash function (SHA-256) as an input. The hash function computes a new resulting hash as an output. This generated hash is then compared to the target value of the network (at the given time). If the hash value is larger than the target value, then the nonce is readjusted and a new input is sent to the hash function to obtain a new potential output. The problem of finding the appropriate hash value that is smaller than the target value is at the heart of PoW, and it can only be solved using brute force. Once a hash value smaller than the target value is discovered by a miner, this hash can then be used in the block header for the candidate block. The first miner to discover the hash is considered to be the winner. The winning miner has shown PoW that she did to discover the hash, so the transactions contained within the block are now considered valid. This block can now be added to the blockchain. Additionally, the winning miner also earns the reward for solving the PoW problem, which is a certain number of Bitcoins. This whole process from packaging transactions into a block, to finding the hash and announcing the block to the Bitcoin network repeats itself approximately every ten minutes.

We introduced some new terminology in Figure 2-2, so let's describe the terms here properly for the sake of completion.

- *Candidate block*: An incomplete block, created as a temporary construct by a miner to store transactions from the transaction pool. It becomes a complete block after the header is completed by solving the PoW problem.
- *PoW*: The problem of discovering a new hash that can be used in the block header of the candidate block. This is a computationally intensive process that involves evaluating a hash taken from the most recent block and appending a nonce to it against the target value of the network. This problem can only be solved using brute force; that is, multiple trials of using the hash (from the most recent block header) and nonce being adjusted each time are necessary to solve the PoW problem.
- *Nonce*: A 32-bit value that is concatenated to the hash from the most recent block header. This value is continuously updated and adjusted for each trial, until a new hash below the target value is discovered.
- *Hash function*: A function used to compute a hash. In the Bitcoin protocol, this function is the SHA-256.
- *Hash value*: The resulting hash output from a hash function.
- *Target value*: A 265-bit number that all Bitcoin clients share. It is determined by the difficulty, which is discussed shortly.
- *Coinbase transaction*: The first transaction that is packaged into a block. This is a reward for the miner to mine the PoW solution for the candidate block.
- *Block header*: The header of a block, which contains many features such as a timestamp, PoW, and more. We describe the block header in more detail in Chapter 3.

■ **Note** After going over the terms defined above, revisit Figures 2-1 and 2-2. Some concepts that were abstract will become clear now and the information will integrate better.

Now that we have a better idea of how mining works, let's take a look at mining difficulty and target values. Those two concepts are analogous to dials or knobs that can be adjusted over the course of time for the network and all Bitcoin clients update to follow the latest values. So what is mining difficulty? Essentially, it can be defined as the difficulty of finding a hash below the target value as a miner is solving the PoW problem. An increase in difficulty corresponds to longer time needed to discover the hash and solve PoW, also known as mining time. The ideal mining time is set by the network to be approximately ten minutes, which implies that a new block is announced on the network every ten minutes. The mining time is dependent on three factors: the target value, the number of miners in the network, and mining difficulty. Let's look at how these factors are interconnected.

1. An increase in mining difficulty causes a decrease in the target value to compensate for the mining time.
2. An increase in the number of miners joining the network causes an increase in the rate at which PoW is solved, decreasing the mining time. To adjust for this, mining difficulty increases and the block creation rate returns to normal.
3. The target value is recalculated and adjusted every 2,016 blocks created, which happens in approximately two weeks.

As we can see, there is a common theme of self-correction in the Bitcoin network that allows it to be very resilient. Miners are the heartbeat of the Bitcoin network and they have two main incentives for participation:

- The first transaction to be packaged in a block is called the coinbase transaction. This transaction is the reward that the winning miner receives after mining the block and announcing it on the network.
- The second reward comes in the form a fee charged to the users of the network for sending transactions. The fee is given to the miners for including the transactions in a block. This fee can also be considered a miner's income because as more and more Bitcoins are mined, this fee will become a significant portion of the income.

Now we can put these concepts together in the form of another flowchart, as shown in Figure 2-3. This will help solidify the process of mining in the context of difficulty and target values.

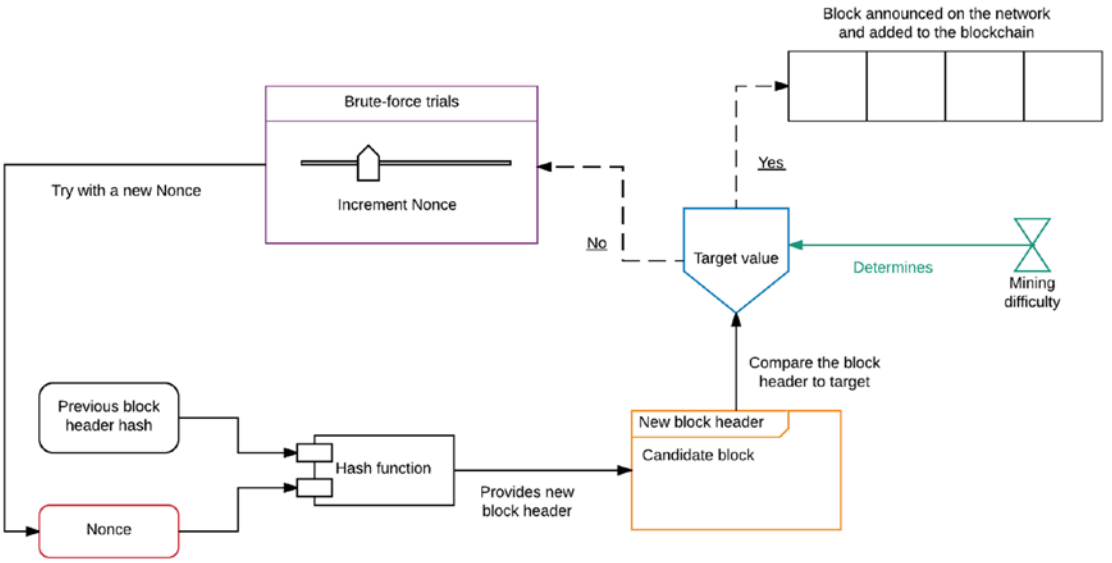


Figure 2-3. *Solving the PoW problem*

Miners across the network compete to solve the problem and the winning miner announces the block to the network, which then gets incorporated in the blockchain. To solve the PoW, a miner has to keep generating new hash values (through the hash function) using the incremented nonce until a hash that is below the target value is discovered. In this case, notice that the nonce is the only adjustable value. This is a simplified PoW scheme and there are small differences in its implementation.

■ **Note** The term *mining* is used because the process is similar to the mining of rare metals. It is very resource intensive and it makes new currency avaliable at a slow rate, just like the miners in the Bitcoin protocol getting rewarded.

We talked about the self-correction properties in Bitcoin network and how they allow the network to adapt. Next, we take a look at an unexpected case of having a very large number of miners in the network as Bitcoin gained popularity. This led to an arms race of sorts and it had far-reaching consequences. First, though, we need to talk about the new types of mining hardware that emerged.

Mining Hardware

As Bitcoin started to gain more popularity and acceptance with merchants, more miners joined the network in hopes of earning rewards. Miners began to get more creative with how they approached mining, such as using specialized hardware that can generate more hashes. In this section, we discuss the evolution of mining hardware as Bitcoin started to spread globally.

- *CPU mining*: This was the earliest form of mining available through the Bitcoin clients. It became the norm for mining in the early versions of the Bitcoin client, but was removed in the later updates because better options became accessible.
- *GPU mining*: This represented the next wave of mining advancements. It turns out that mining with a graphics processing unit (GPU) is far more powerful because it can generate hundreds of times more hashes than a central processing unit (CPU). This is now the standard for mining in most cryptocurrencies.
- *FPGAs and ASICs*: Field-programmable gated arrays (FPGAs) are integrated circuits designed for a specific use case. In this case, the FPGAs were designed for mining Bitcoins. The FPGAs are written with very specific hardware language that allows them to perform one task very efficiently in terms of power usage and output efficiency. Shortly after the introduction of FPGAs, a more optimized, mass-produceable, and commercial design came out in the form of application-specific integrated circuits (ASICs). The ASICs have a lower per-unit cost, so the units can be mass produced. The ASICs-based devices are also compact, so more of them can be integrated in a single device. The ability of ASICs to be combined in arrays at a low price point made a very convincing case for accelerating the rate of mining.
- *Mining pools*: As the mining difficulty increased due to the rise of ASICs, miners realized that individually, it was not financially wise to continue mining. It was taking too long, and the reward did not justify the resources that went into mining. The miners then organized themselves into groups called pools to combine the computational resources of all the members and mine as one unit. Today, joining a pool is very common to get started with mining in almost every cryptocurrency.
- *Mining cloud services*: These are simply contractors who have specialized mining rigs. They rent their services to a miner according to a contract for a given price to mine for a specific time.

It is easy to see how ASICs completely changed the mining game after developers and hardware hobbiysts realized that custom arrays of ASICs can be assembled at a fairly cheap price point. It was the beginning of a kind of arms race in Bitcoin hardware, as developers were designing new chips and buying new equpiment for mining rigs that allow them to mine the most Bitcoin. This initial push, driven by profit, accelerated Bitcoin's reach and created a golden era for the alternative currency. More developers and enthusiasts joined in, buying custom hardware to maximize their profits. The network responded by increasing the difficulty as the number of miners increased. Within a short time span, the bubble could not sustain itself for the miners due to the self-correcting features present in the protocol and the difficulty kept rising. In some cases, the hardware that miners purchased could no longer mine profitably by the time it arrived from the factory. A significant capital investment was required up front to achieve any appreciable

returns. Most of the ASICs hardware is now historic, and even Bitcoin mining pools are not profitable for the average miner. The startups and companies that commercialized ASICs and custom hardware made a decent short-term profit and then flopped. We examine a few of those massive failures in the next section.

Startup Stories

In this section, we highlight a few stories from the gold rush era of Bitcoin between mid-2013 and late 2014. The startups covered here followed the strategy of selling pickaxes to make profits, but some took it a step further. The first startup we discuss is Butterfly Labs. This company out of Missouri came about in late 2011 with the promise of selling technology that was capable of mining Bitcoin leaps and bounds ahead of the competition. Their ASICs were supposedly able to mine Bitcoin 1,000 times faster, and they opened up for preorders soon after the initial announcement in 2012. Miners flocked to purchase the hardware, which was promised for delivery by December of the same year. Butterfly Labs collected somewhere between $20 million and $30 million in preorders as reported by the Federal Trade Commission (FTC). Shipments started to roll out to only a few customers around April 2013, but most customers did not receive their mining equipment for another year. When the customers did recieve their machines, they were obsolete, and some accused Butterfly Labs of using the hardware to mine for themselves before delivering them. Despite being unable to follow through with their initial orders, Butterfly Labs began offering a new and much more powerful miner and opened preorders for that new miner. Ultimately, the company became one of the most hated in the Bitcoin community, and the FTC had to step in to shut it down.

The second company we discuss, CoinTerra, is a more complicated case because the startup was founded by a team that had deep expertise in the field. The CEO, Ravi, was a CPU architect at Samsung previously, and the company's board included many other leaders in the field. Initially, they were venture backed and well funded, and in 2013, they announced their first product, TerraMiner IV, which was supposed to be shipped in December of the same year. The company could not ship the product in time and eventually pushed the date back. The miner still did not arrive in 2014 and eventually CoinTerra apologized to customers, offering them some compensation, which was also largely delayed, frustrating customers even further. It seems that the company is trying to pivot to cloud mining services, but most of its customer base has already lost their trust.

The final case focuses on a startup called HashFast. Similar to previous two examples, HashFast was offering miners called Baby Jet that would be delivered in December 2013. The team at HashFast overpromised the features and underdelivered at a time when difficulty skyrocketed. It is likely that the company took the cash from early adopters to fund its own development, and when they encountered difficulties, the customers demanded refunds for their orders. The problem at the time was that the price of Bitcoin increased steadily, so the company did not have enough funds to pay back the customers. They were facing multiple lawsuits and running out of cash reserves very fast. Eventually, a judge allowed the auctioning of all assets that the company owned to pay back the creditors and investors. A common theme these companies share is that they were frequently unable to deliver mining hardware on the promised timeline and significantly delayed or refused to issue any refunds to their customers.

We can construct a general scheme of operations from the cases presented here and other ASICs startups that failed similarly to Butterfly Labs:

- Open for preorders at very high prices and falsely advertise a ridiculously high hashing rate with a huge return on investment.
- Invest all the funding from preorders to begin research and development for ASICs and custom hardware.
- Once the mining hardware has been obtained from overseas manufacturers, use it to mine nonstop for months internally.

- Broadcast to customers through social media that the manufacturing process is taking longer than expected.
- Deliver the hardware only to the customers that threaten to sue as early proof that shipments have begun rolling out.
- Deliver the ASICs hardware to other customers when it is already severely out of date.
- Customers complain and file lawsuits, and the company eventually falls apart and faces huge fines.

New Consensus

We conclude this chapter by talking about the same theme that we started it with: consensus. This chapter's central idea was that in Bitcoin, mining is used to reach consensus to prevent users from double spending and validate all the transactions. However, since the advent of Bitcoin, other consensus algorithms have been developed. We refer to the PoW algorithm referenced in the original Bitcoin protocol for reaching consensus as the Nakamoto Consensus. A new consensus algorithm that has recently become popular is known as proof of stake (PoS), where the participants essentially play the role of validators. In Bitcoin, bad actors with fraudulent transactions have to face a rigorous approval process and validation from the network of miners. In PoS, the participants have a stake in the network (hence the name) in the form of currency. As such, they want to see the network succeed and trust emerges in blocks that have the largest stake of currency invested by the validators. Additionally, the malicious validators will get their stake slashed for acting in bad faith. We dive into the technical aspects of PoS, and how it compares to the mechanism of PoW, later in the book. Our journey ends in this chapter with consensus and we pick up our discussion on the Bitcoin network and the blockchain in the next chapter.

Summary

In this chapter, we talked about the concept of mining and presented the technical background necessary to understand how miners verify blocks. We discussed in depth the backbone of mining in Bitcoin called PoW, and throughout the remainder of the book, we present other consensus mechanisms. Then, we described the arms race in Bitcoin mining over producing the best hardware, which led to the huge rise in difficulty, and the startup failures that resulted from that time period. Finally, we ended the chapter with a mention of PoS, which we return to in a later chapter.

References

The key references used in preparing this chapter were Michael Nielsen's post (http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/) on Bitcoin mining, and Aleksandr Bulkin's post (https://keepingstock.net/explaining-blockchain-how-proof-of-work-enables-trustless-consensus-2abed27f0845). The remaining references can be found at the end of the book.

CHAPTER 3

Foundations of Blockchain

You never change things by fighting the existing reality.
To change something, build a new model that makes the existing model obsolete.

—R. Buckminster Fuller

The blockchain is a decentralized data structure with internal consistency maintained through consensus reached by all the users on the current state of the network. It's an enabling technology that resolved the Byzantine generals' problem (message communication between untrusted parties) and opened up a new horizon of possibilities for trustless transactions and exchange of information. If the Internet democratized the peer-to-peer exchange of information, then the blockchain has democratized the peer-to-peer exchange of value. We begin this chapter by exploring how transactions work between users on the Bitcoin network. This entails a technical discussion of structures of a block and a transaction. We then dive into the role of wallets and user addresses. After talking about wallets, we shift our focus to Simple Payment Verification (SPV) implemented in the Bitcoin network. SPV allows us to understand why blocks have a peculiar structure and more important, how the Bitcoin network can retain efficiency despite the network scaling at a high rate. Finally, we conclude our discussion by talking about hard and soft forks in the blockchain. We present the implications of forks in the context of forward compatibility for merchants and users involved in running the Bitcoin-core code.

Transaction Workflow

The central purpose of the Bitcoin protocol is to allow transactions to occur over the network between users in a decentralized manner. We have been talking about small fragments of the protocol to build up background. Now we can integrate those concepts into a single framework and explore the blockchain. The ultimate result of mining is increasing the number of blocks as the network evolves over time. To understand how transactions occur between two users (Alice and Bob), we first need to understand the structure of blocks that hold the transactions. In the simplest terms, the blockchain is a collection of blocks bound by two main principles:

- *Internal consistency*: There are a few design principles inherent to the functioning of each block that make the blocks internally consistent. For instance, each block links to the previous one in the chain and has a creation timestamp. Such mechanisms in the blockchain allow it to be an internally coherent data structure that can keep a consistent record of transactions.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_3

- *Consensus on transactions*: The concept of mining described in Chapter 2 is just one implementation for verifying transactions; there are different methods where no brute-force hashing is involved. However, in every one of these implementations, there is a scheme of reaching consensus on the transactions that have transpired during some interval in the network. We can generalize this verification of transactions for a decentralized system by either using some sort of PoW or a similar strategy that pools transactions that are then checked by users on the network.

A transaction is essentially a data structure carried on a block, but how exactly? To discover the process, let's look at the complete structure of a block in Figure 3-1. Each block has at least two unique components: the block header, which contains a unique hash (called the merkle root) that uniquely identifies a block, and the transaction list, which contains new transactions. Note that each block contains the same amount of transactions in the list, but the precise transactions between users are different. This is because only one block wins the mining race every ten minutes on the blockchain, other candidates are rejected, and the race starts again. In this simplified model, there are only two other components of a block: the block size, which is kept consistent for the entire network, and a counter for the number of transactions in each block. Here, we focus more on the block header and the transaction list.

The block header contains a few standard components, such the as difficulty target and the nonce discussed previously. It also contains the version number of the Bitcoin core-code that the winning miner is running. The timestamp is also a unique feature of every block, as it unmistakably identifies one particular block in the network. The header also contains a hash from the previous block in the chain, and a special hash that identifies this block called the merkle root. We discuss how this special hash comes to be later in this chapter.

■ **Proof of life** Recently, there were rumors that Julian Assange, the WikiLeaks founder, had died. Assange recently did an Ask Me Anything session on Reddit and responded to the rumors by reading the most recent block hash from the blockchain to prove that he was indeed alive. The block was created only ten minutes earlier, so this could not have been a prerecording, thus proving beyond any shadow of a doubt that Assange was alive. This was the first time the block hash found a use in a sense of popular culture, and Assange called it a proof of life.

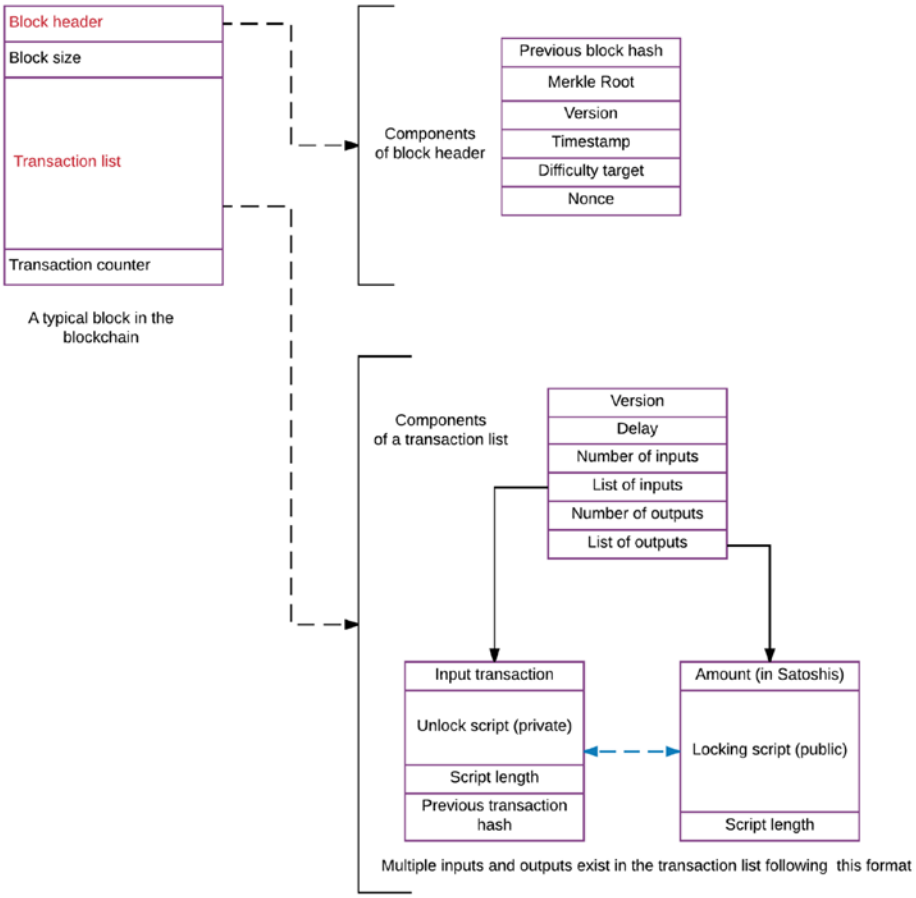


Figure 3-1. *Simplified overview of the structure of a block*

The block header and transaction list are the two components that stay unique to every block. The block header is made up of several smaller parts, the most peculiar of which is the merkle root, a hash that uniquely identifies a block. The header contains the hash of the previous block, the nonce used to create that particular block, and the difficulty of the network. These are standard mining components that we discussed previously. Each block also contains a list of transactions. Aside from the actual transactions, the transaction list also contains a few components that are crucial to how a block will accept the transaction. For instance, the lock time delay dictates when a transaction can be accepted into a block. Finally, the list contains all the transactions accepted into this block as a series of signed inputs and outputs that ensure the transfer of Bitcoins from the sender to the receiver.

There are several new terms and concepts introduced here and we will go through all of them now. We already talked about the block header and the concepts of timestamp on a block, the merkle root, and a hash from the previous block. Now we focus on the components of the transaction list. Let's begin with the delay. The proper technical term is lock time delay, which refers to the time after which a transaction can be accepted into a block. The precise mechanism involves the use of a parameter called blockheight, which increases as more blocks are added to the blockchain. A given transaction remains locked and unverified until the blockheight specified for that transaction is exceeded.

Next is the concept of transaction inputs and outputs. The foundational unit of a transaction is called an unspent transaction output (UTXO), which can have a value given in Satoshis. Similar to how a dollar can be split into 100 cents, Bitcoin can be divided into an eight-decimal unit called Satoshis. A UTXO is a unit of currency controlled by users (the users are also known as owners) and recorded on the blockchain. More precisely, UTXO is really the currency balance, or the unspent currency present in the Bitcoin economy. The entire network accepts UTXO as currency and whenever a user receives Bitcoin, the amount is recorded on the blockchain as a UTXO. Essentially, the Bitcoin belonging to a user can be spread across several transactions and many blocks as UTXO. As a result, there is no defined concept of stored balance for a particular user, but only UTXOs spread across the network possessed by the owners. The idea of an account balance is actually created for a user by the wallet software, which searches the blockchain and collects all the UTXO belonging to a particular address. We discuss the concepts of wallets and addresses shortly.

To understand UTXO properly, we need to talk about the concept of change. The idea is very simple, actually. Think about the last time you bought groceries and paid with cash. You probably got some change back that was left over from your payment. UTXOs have a similar concept, as shown in Figure 3-2. Every transaction is split into a portion that is spent and locked (assigned) to another user, and a portion that gets returned back to the original user, just like the change that you would get from shopping. In a transaction, UTXOs that are consumed by the transaction are called the inputs, and the UTXOs created by a transaction are called the outputs. The example in Figure 3-2 illustrates a similar scenario, where Bob wants to send one BTC to Alice, but in the process, the ten BTC owned by Bob are split into two parts: the one BTC sent to Alice, which is now assigned to her, and the nine BTC that are returned to Bob in the form of UTXOs. Both of these components are recorded on the blockchain because they are a part of a transaction, as shown in Figure 3-2.

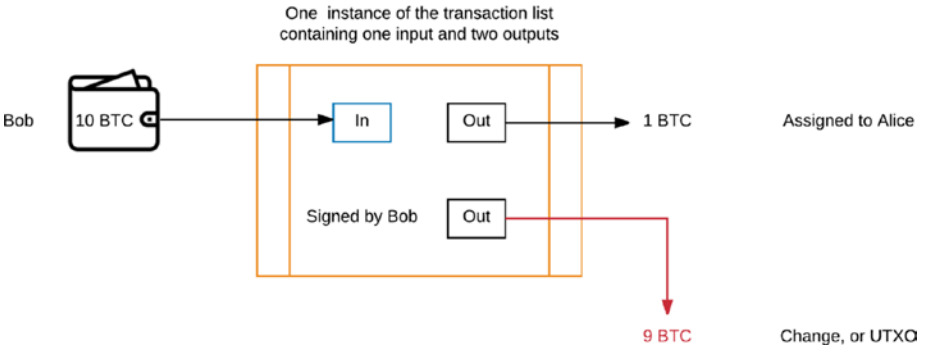


Figure 3-2. *Format of a UTXO in the transaction list*

In this example, Bob wants to send one BTC to Alice, and Figure 3-2 shows how this transaction occurs. The BTC owned by Bob is used as the input of the transaction and the output is two parts, one sent to Alice for one BTC and the second one returned as change back to Bob. It should be noted here that the initial transaction, the newly assigned transaction, and the change are recorded on the blockchain as the input and output.

Now that we have a better grasp of UTXOs, let's talk about how transactions are assigned from one user to another. This involves the use of private–public key pairs that lock and unlock the transactions. The process works as follows:

- A user, Alice, initiates a transaction that she wants to send to Bob.
- Alice uses her private key to sign the transaction.
- The transaction is broadcast on the network and anyone can use Alice's public key to verify that the transaction originated from her.

- Bob receives the transaction after it has been verified on the network and propagated to him.
- Bob unlocks the transaction using his private key. The transaction was signed with a script such that only the recipient could unlock the transaction and assign it to themselves.

We mention that the transaction locking and unlocking mechanisms use a script, so what is this script? The Bitcoin protocol uses a minimal, bare-bones, Turing-incomplete programming language to manage transactions. Satoshi's intention was to keep the programming logic very simple and largely off the blockchain whenever possible. A script is attached to every transaction and it contains instructions on how the user receiving Bitcoins can access them. Essentially, the sender needs to provide a public key that anyone on the network can use to determine that the transaction did indeed originate from the address contained in the script, and a signature to show that the transaction was signed by using the sender's private key. Without the private–public key pair authorization, transactions between users would not occur. Let's complete the picture that we started to create with the UTXOs, shown in Figure 3-3.

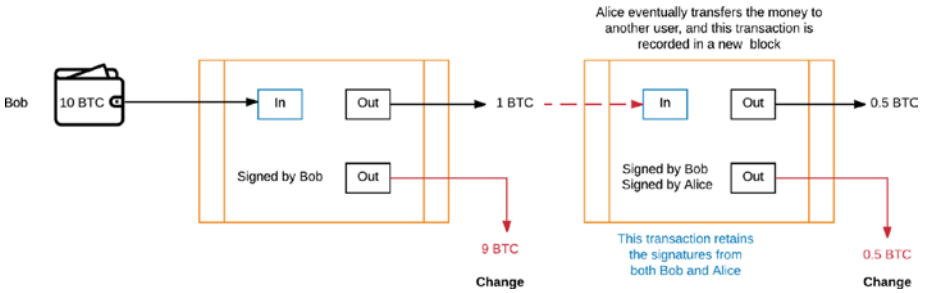


Figure 3-3. *Transactions on the blockchain*

Conceptually, it might be bizarre to consider transactions on the network as UTXOs being spread across hundreds of blocks, but this process is exactly how transactions are propagated across the network. In this example, Bob first initiated the transaction that was sent to Alice in which one BTC was assigned to Alice. He received nine BTC in change as the unspent output. Alice further sends 0.5 BTC to another user and in doing so, she receives 0.5 back in change from her transaction. Notice that the first transaction was signed by Bob, who initiated the transaction, and then Alice signed the second transaction. In a sense, the output from the first transaction became an input for the second, so Bob's signature was retained as proof of the first transaction and Alice's signature now serves as the unlocking mechanism. This is how transactions can be tracked across the Bitcoin network from the origin to the final owner (final address). By using network addresses, the network retains a level of pseudonymity.

Now that we have talked about UTXOs, signatures, scripts, and how transactions are recorded, let's integrate these concepts and review the workflow of a transaction between Alice and Bob, shown in Figure 3-4.

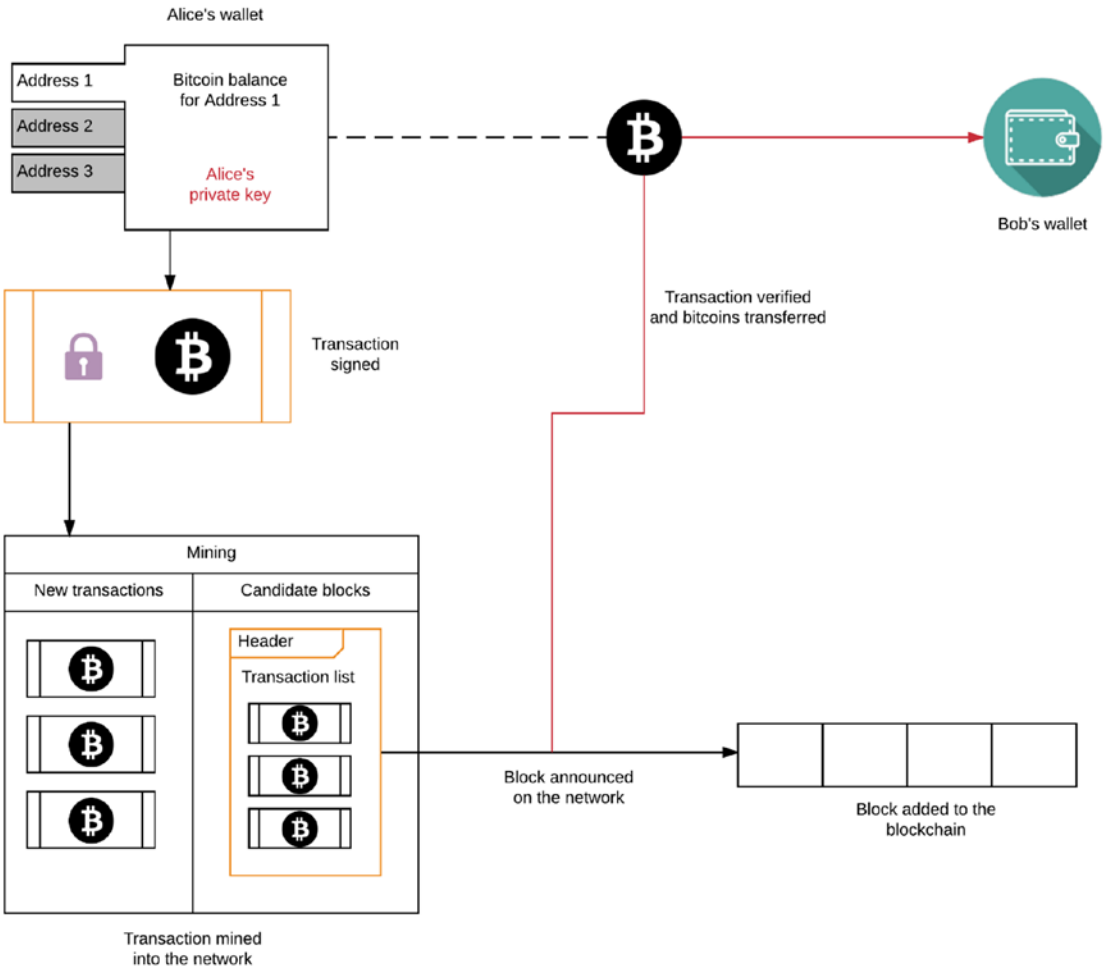


Figure 3-4. *Overview of a transaction on the network*

Alice initiates the transaction from her wallet, which contains multiple addresses. Each address has a certain amount of Bitcoin balance (the sum of all UTXOs associated with that address) that can be used to create new transactions. The transaction is then signed using Alice's private key and it enters the mining phase, where it will be packaged into a candidate block. As the mining concludes, the winning miner announces the block on the network and the block is included into the blockchain. The remaining transactions are thrown into the pool of transactions to be added. The transaction propagates to Bob, who can now use his private key to unlock the transaction output amount and use it. The ideas of UTXOs, signing, and script locking and unlocking provide deeper insights into how the blockchain remains internally consistent as a decentralized ledger.

Figure 3-4 introduces the concept of the wallet, which can be used to initiate transactions. Wallets are now a standard part of the Bitcoin core-code and they mainly serve three purposes for the users:

- *Create transactions*: A user can create transactions easily with a graphical interface using the wallet.
- *Maintain balance*: The wallet software tracks all the UTXOs associated with an address and gives a user his or her final balance.
- *Maintain multiple addresses*: Within the wallet, a user can have multiple addresses and each address can be associated with certain transactions.

In a sense, addresses are the only means of ownership in the Bitcoin network. UTXOs and balances are associated with a particular address and a user can create as many addresses as her or she wants. We saw in Figure 3-4 that Alice had three addresses in her wallet, and each of the addresses can work with her private key. There are actually other types of wallets, aside from a software wallet. Figure 3-4 used a software wallet, but the process is similar for the other two main types, mobile wallets and a cold storage physical wallet.

Mobile wallets have largely been designed for the sake of convenience and as a gateway into the world of mobile payments using cryptocurrencies such as Bitcoin. These wallets often serve as an independent but miniaturized version of a complete wallet, and allow for access to balances and transactions on the go. The apps that work as wallets are often designed in an open source environment, so they are also helping bring developers and power users together in the community. Cold storage wallets are a more permanent method of storing Bitcoins over a long period of time. There have been instances where wallets got corrupted or the users couldn't remember the key to unlocking those wallets, rendering their balance effectively useless. There is no recovery mechanism for a password on a wallet. The idea here is to create a new wallet, and send a transaction to a new address on that wallet. Now this wallet can be backed up and saved to a physical device such as a flash drive and stored away securely. Once that transaction has been verified on the blockchain, your Bitcoins are safe to be retrieved from the flash drive at any time. This can be done to prevent any accidents from happening and to keep your currency separate from the main wallet that you use to conduct transactions or mine for Bitcoins. Some developers have taken a step further and created paper wallets where the address is encoded in a QR code and a private key for that particular wallet is also printed on the paper in another QR code.

■ **Note** How can you actually see your transaction taking place on the Bitcoin network without having to write a script or code yourself to do it? In Bitcoin (and most cryptocurrencies), there is a feature called Blockchain Explorer, usually a web site where all transactions are visible from the Bitcoin network. You can obtain all sorts of details about transactions, such as the origin of the transaction, the amount, the block hash, or how many verifications it received.

Simple Payment Verification

So far, we have talked about the structure of blocks, transaction lists, how transactions occur between users, and how they are recorded on the blockchain. Blocks are fundamentally data structures linked on the blockchain, and transactions can be thought of as property of that data structure. More precisely, in the case of blockchains, transactions are represented as leaves of a merkle tree. Hashes have been used throughout the Bitcoin protocol as a method for maintaining data consistency because a hash is very easy to verify and nearly impossible to reverse engineer. Building on these properties, we can tackle a very difficult technical challenge on the blockchain: How can we check if a particular transaction belongs to a block?

Checking through an *N* number of items in a list would be very inefficient, so we cannot simply check every transaction in a blockchain containing millions of blocks to verify. This is where a merkle tree provides speed and efficiency.

To visualize a merkle tree, refer to Figure 3-5. It is constructed from the transactions of a block to allow fast access for verification purposes. Let's follow the example shown in Figure 3-5. In this case, there are eight transactions collected in a block and represented on a merkle tree. The lowest level is the transactions themselves, and they are abstracted to a higher level by hashing two transactions together and obtaining an output hash. This hash is combined with a second one and hashed again to abstract a higher level. This process is repeated until only two hashes are left. Notice that each level contains information about the level below, and finally the highest level holds a hash with information from the entire tree. This hash is called the merkle root. How would a merkle root assist in finding a transaction? Let's run through the example shown in Figure 3-6 and try to find transaction 6 from the merkle tree. For starters, the merkle root allows us to skip the other half of the tree, and now our search is limited to transactions 5 through 8. The hashes guide the search further, allowing us to step into (reach) transaction 6 in just three steps. Compare this to searching through the whole tree, stepping into every level, and comparing every transaction to see if it is indeed transaction 6. That process would be more involved in terms of the steps taken and the time needed, and this becomes too exhausting if the search expands to millions of transactions.

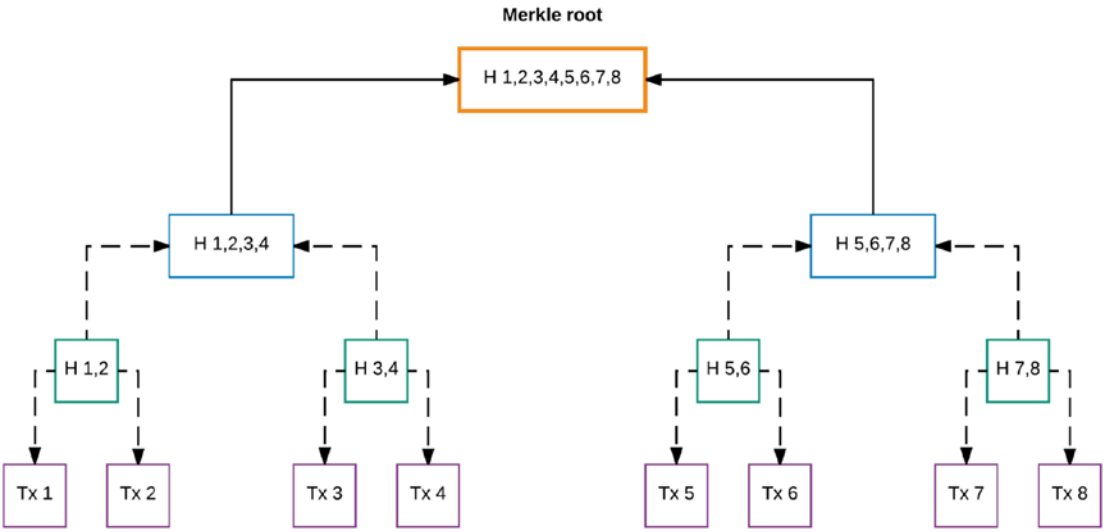


Figure 3-5. *Constructing a merkle root*

The lowest level is formed from the transactions and the general idea is to keep hashing two elements together and retain some information about the level below. Ultimately, we are left with only two elements that are hashed together to form the merkle root.

When would searching for a transaction come in handy? Every new user to get started with the standard Bitcoin wallet client has to download the entire blockchain. Over time, the blockchain has increased in download size, recently reaching a few gigabytes. This can be intimidating to new users, who cannot use their wallets until the blockchain download is finished, and it might turn them away. To solve the problem of having to download a bloated blockchain with historic transactions, Satoshi came up with a solution called SPV. The rationale in SPV is to create a wallet client that downloads only the block headers instead of the entire blockchain. This new lightweight client can use the merkle root in the block headers to verify if a particular transaction resides in a given block. The precise mechanism requires the wallet to rely on a merkle

branch and reach the specific transaction, much like the example shown in Figure 3-6. Currently, for Bitcoin, there is an alternative wallet client known as Electrum that implements SPV and allows new users to avoid the hassle of downloading the entire blockchain.

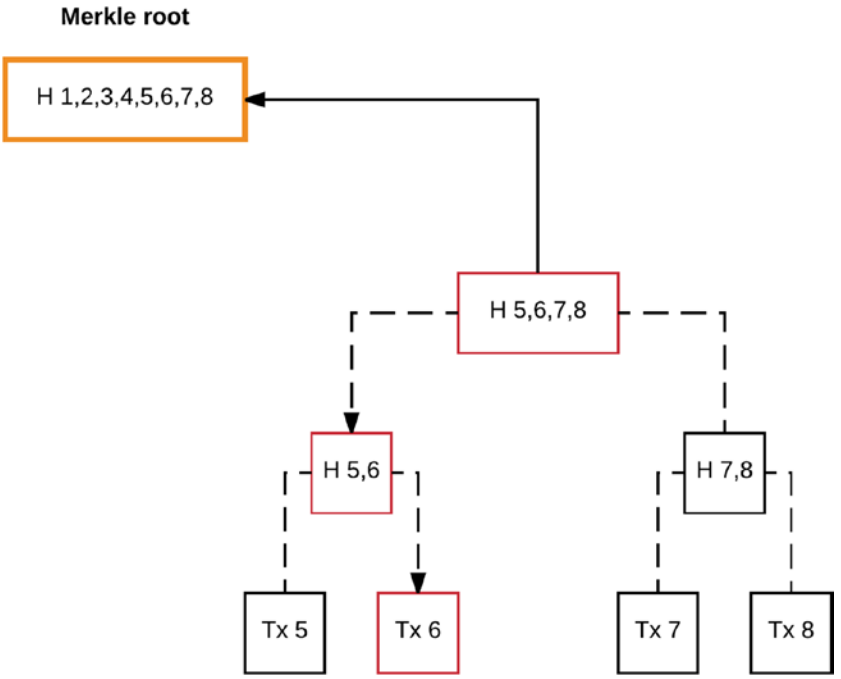


Figure 3-6. *Finding a transaction using the merkle root*

The root allows us to skip half of the tree during our search and the next level narrows down the search even further. Using the merkle root, we can reach the transaction in just three steps, which allows a very high operational efficiency that we would need in Bitcoin's current network. The path to reaching transaction 6 is also known as a merkle branch, connecting the root to a leaf.

Blockchain Forks

Here's an interesting scenario to consider: Several miners are competing to solve the PoW and create a block. Incidentally, two miners find a valid value within a few seconds of each other and broadcast them to the network. What happens now? This situation is known as a fork, and it is completely normal on the Bitcoin network, especially as the network starts to scale and include thousands of miners. To resolve the fork, there are a few rules in place on the network, called the consensus rules. The tie creates two versions of the blockchain, and this tie is resolved when the next block is discovered. Some of the peers will be working on one version of the blockchain, and others will be working on the second version. When the next block is discovered, one of the chains will become longer due to the inclusion of this new block. This chain now becomes the active chain and the nodes will converge to the new chain. This process is visually illustrated in Figure 3-7.

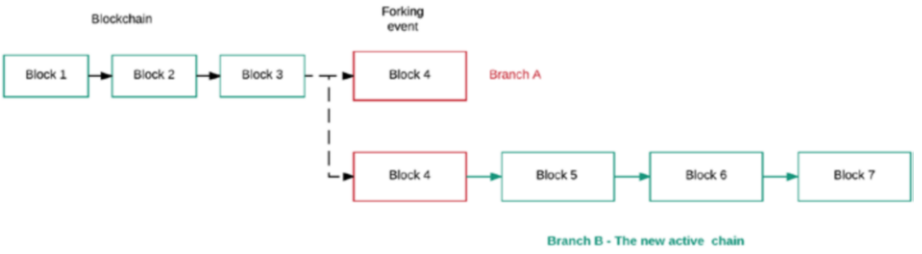


Figure 3-7. *Fork in the chain*

In this example, block 4 is discovered at the same time by two miners, but the tie is resolved when the next block is discovered on Branch B. This branch now becomes the active chain and all the nodes converge to using Branch B as the new active chain.

Normal forks on the blockchain are not a concerning event, because they are usually resolved within a matter of minutes. Soft and hard forks are an entirely different matter, however. These can occur in the case of upgrades to the Bitcoin core-code where a permanent split happens between nonupgraded nodes that cannot validate any newly created blocks and upgraded nodes that have begun creating blocks following the new consensus rules. Two entirely different types of blocks begin to appear on the network and the network is unable to converge on a single active chain until the nodes are upgraded to the new rules.

In this case, there are two possible outcomes. The first possibility is that the majority of the network switches over to the new rules (a soft fork), and the new rules allow for the carryover of some portion of the valid old blocks. The second alternative is that the old blocks remain invalid for the new nodes, and no old blocks are accepted in the network by the new nodes. This is a hard fork, where no forward compatibility exists and the old blocks will no longer be accepted by the new nodes. All the miners and nodes have to upgrade to the new software so that their blocks can be considered valid under the new rules. A hard fork can be chaotic and cause a problem for users and merchants that have created payment terminals and interfaces relying on the old rules for transactions. They have to upgrade their back-end software to be compatible with the new rules and ensure a smooth transition of incoming Bitcoins. A hard fork is not upcoming for the Bitcoin network, but developers have begun researching just how complex the process might be. We end our discussion of the blockchain forks here, but we return to it later. In the next chapters, we take a look at circumstances in which a hard fork might become necessary in the next generation of Bitcoin protocols.

Summary

In this chapter, we integrated the concepts of mining into the whole blockchain network. We described what a blockchain is and how it functions at a technical level. Then, we described the workflow of a transaction and tracking unspent transaction outputs. We talked about how transactions are put together and propagated on the blockchain and also mining software such as a wallet and mining client. Then, we put mining in the context of a proper network and showed how a transaction goes from being included in a block to being propagated. After that, we talked about the concept of SPV and the importance of merkle hashes and roots in Bitcoin. We ended the chapter with a discussion of blockchain forks and how they influence the network, which we revisit later in the book as well.

References

The main references used to prepare this chapter are the Bitcoin Developer Guide for discussions on UTXOs and block headers. Georg Becker's work on Merkel Tree signatures was used to prepare the sections on Simple Payment Verification and Merkel roots.

CHAPTER 4

Unpacking Ethereum

Any sufficiently advanced technology is indistinguishable from magic.

—R. Buckminster Fuller

Ethereum is an open source, decentralized, blockchain platform with computational capabilities that reconstruct elementary currency exchange into a transfer of value between users via a scripting language. Ethereum is widely recognized as a successor to the Bitcoin protocol, generalizing the original ideas and enabling a more diverse array of applications to be built on top of blockchain technology. Ethereum has two essential components. The first is a Turing-complete virtual processor that can load resources and execute scripts, called the Ethereum Virtual Machine (EVM). The second component is a token of value called Ether, which is the currency of the network used for user-to-user transactions or compensation to miners of the network. In this chapter, we begin our journey with an overview of Ethereum's architecture in comparison to Bitcion, focusing on the EVM and Turing-completeness properties. Following the architecture, there is a short discussion on the accounts model in Ethereum, and account representation with Merkel-Patricia Trees. This will lead us to global state in Ethereum, account storage and gas, which is a spam-prevention mechanism in the network. Then, we deconstruct the notion of a smart contract enabled by EVM, the security concerns revolving around sandboxing executable code, and how the EVM pushes executable code (bytecode) to the blockchain. After that, we provide an introduction to Solidity, a programming language for writing smart contracts in Ethereum. We will explore the syntax of Solidity, and the common integrated development environments (IDEs) being used to work with it. Next, we focus on the World Computer model proposed using Ethereum and a few related decentralized technologies such as IPFS and Whisper. Then, we will look at the apps available in Ethereum. On the enterprise side, a particularly noteworthy development is the Blockchain-as-a-Service (BaaS) deployed on the Azure cloud by Microsoft. For the network, distributed apps (Dapps) are being built on top of Ethereum and published using other World-Computer components such as Mist.

Overview of Ethereum

It was around mid-2013 when a majority of the Bitcoin community was starting to flirt with the idea of other applications beyond simply currency. Soon, there was a flood of new ideas discussed in online forums. Some common examples include domain registration, asset insurance, voting, and even Internet of Things (IoT). After the hype started to fade away, a more serious analysis of the Bitcoin protocol revealed severe limitations of potential applications that can be built on top of the blockchain.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_4

A crucial point of debate was whether a full scripting language should be allowed within the blockchain or applications should be built with logic residing outside of the blockchain. There were two key issues that sparked this debate:

- The scripting language and OPCODES in the Bitcoin protocol were designed to be very limited in functionality.
- The protocol itself was not general enough, and alternative currencies such as Namecoin and others emerged specialized for one specific task. The big question at the time was this: How can a protocol be generalized such that it becomes future-compatible with applications that we know nothing about?

Eventually, two schools of thought emerged regarding scripting: Traditionally, Satoshi's paper proposed keeping the scripting language very limited in functionality. This would avoid the security concerns of having executable code in the blockchain. In a sense, the blockchain executable code is limited to a handful of necessary primitives that update the distributed states. The second school of thought was championed by Vitalik, who thought of the blockchain as more than just a ledger. He envisioned the blockchain as a computational platform that can execute well-defined functions using contracts and arguments. The design of EVM allows for complete isolation of the executable code and safe execution of the applications built on top of the EVM. Let's begin with the design principles and the core idea behind Ethereum.

■ **Core idea** Instead of building a platform to support specific applications, in Ethereum, we build to support a native programming language with extensibility to implement business logic on the platform using that language.

We return shortly to discuss the implications of this principle. In the meantime, let's talk about another feature of Ethereum, consensus. We discussed the concept of consensus in earlier chapters: In PoW-based cryptocurrencies such as Bitcoin, the network awards miners who solve cryptographic puzzles to validate transactions and mine new blocks. Ethereum uses a different consensus algorithm called PoS. In a PoS algorithm, the validator or creator of the next block is chosen in a pseudorandom manner based on the stake that an account has in the network. Therefore, if you have a higher stake in the network, you have a higher chance of being selected as a validator. The validator will then forge the next block and get a reward from the network. Here, the validator is truly forging a block (in the blacksmith sense of the term) instead of mining, because in PoS, the idea of hardware-based mining has been replaced by this virtual stake. To some extent, the rationale behind using PoS was due to the high-energy requirements of PoW algorithms that became a frequent complaint. Peercoin was the first cryptocurrency to launch with PoS, but more prominent recent PoS implementations can be seen in ShadowCash, Nxt, and Qora. The main differences between Bitcoin and Ethereum as protocols are highlighted in Figure 4-1.

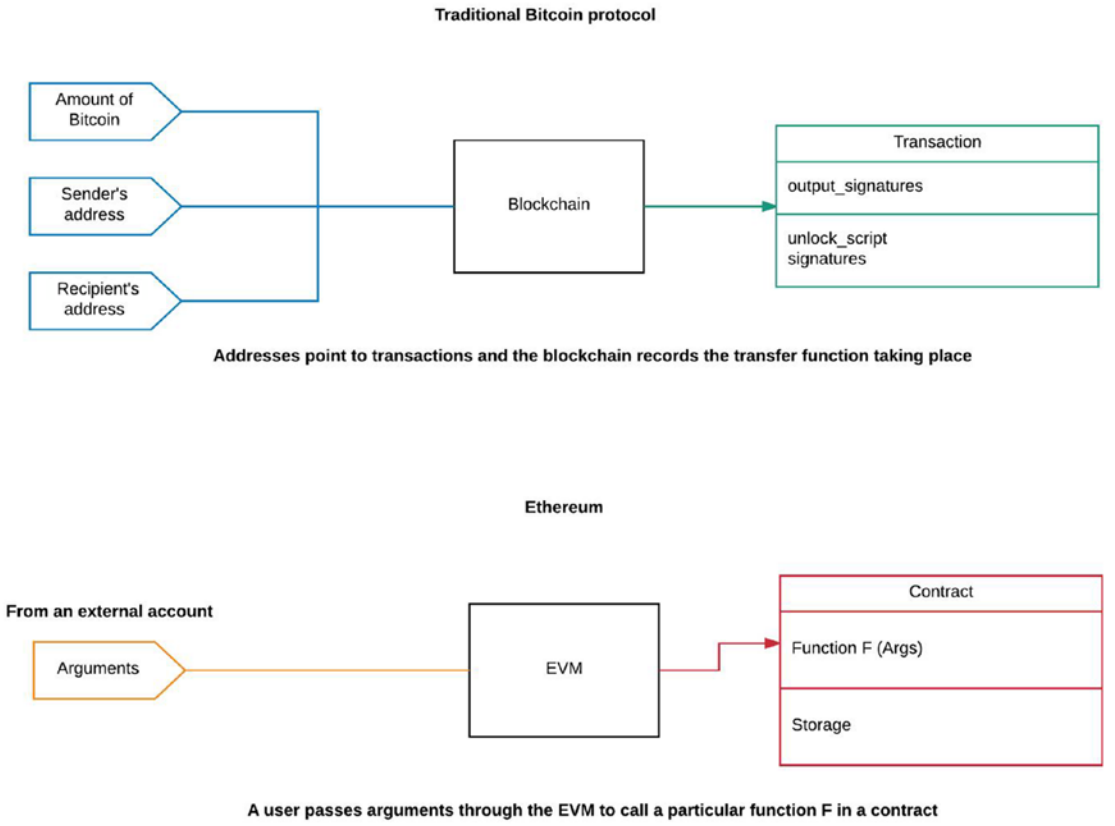


Figure 4-1. *Overview of Bitcoin and Ethereum as computational platforms*

In the Bitcoin protocol, addresses map the transactions from sender to receiver. The only program that runs on the blockchain is the transfer program. Given the addreses and the key signature, this program can transfer money from one user to another. Ethereum generalizes this concept by placing an EVM at every node so that verifiable code can be executed on the blockchain. Here, the general scheme is that an external account will pass arguments to a function and the EVM will direct that call to the appropriate contract and execute the function, granted the appropriate amount of Ether and gas are supplied. As a consequence, every transaction in Ethereum can be considered a function call**.** The function calls and transactions in Ethereum comply with PoS, which has a faster resolution time than the Bitcoin blockchain that relies on PoW. The security level of this process verified by the network is also very high.

Accounts in Ethereum

Accounts are a metastructure in Ethereum and the fundamental operational unit of the blockchain. Accounts also serve as a model to store and track information on the users in the network. There are two types of accounts available on the network.

- *User accounts*: These are user-controlled accounts also known as external accounts. These accounts have an Ether balance, are controlled by public–private key pairs, and can send transactions, but have no associated code. All actions in the Ethereum network are triggered by transactions intiated by external accounts. In the Bitcoin protocols, we referred to these simply as addresses. The key difference between accounts and addresses is the ability to contain and execute generalized code in Ethereum.
- *Contracts*: This is essentially an account controlled by its own code. A contract account is the functional programmatic unit in Ethereum that resides on the blockchain. This account has an Ether balance, has associated code, can execute code when triggered by transactions recieved from other accounts, and can manipulate its own persistent storage. (Every contract on the blockchain has its own storage that only it can write to; this is known as the contract's state.) Any member on the network can create an application with some arbritary rules, defining it as a contract.

If accounts play such a key role, how are they represented on the blockchain? Accounts become an element of the merkle trees, which in turn are an element of every block header. Ethereum uses a modified form of the binary merkle trees called Merkle-Patricia trees. A complete explanation of the Merkle-Patricia tree (see http://www.emsec.rub.de/media/crypto/attachments/files/2011/04/becker_1.pdf) would be beyond the scope of this text, but a graphical synopsis is provided in Figure 4-2.

■ **Note** The two-account system explained here might not remain in Ethereum for the long term. Recently, there has been a push in the development community toward a one-account model, where user accounts are implemented by using contracts.

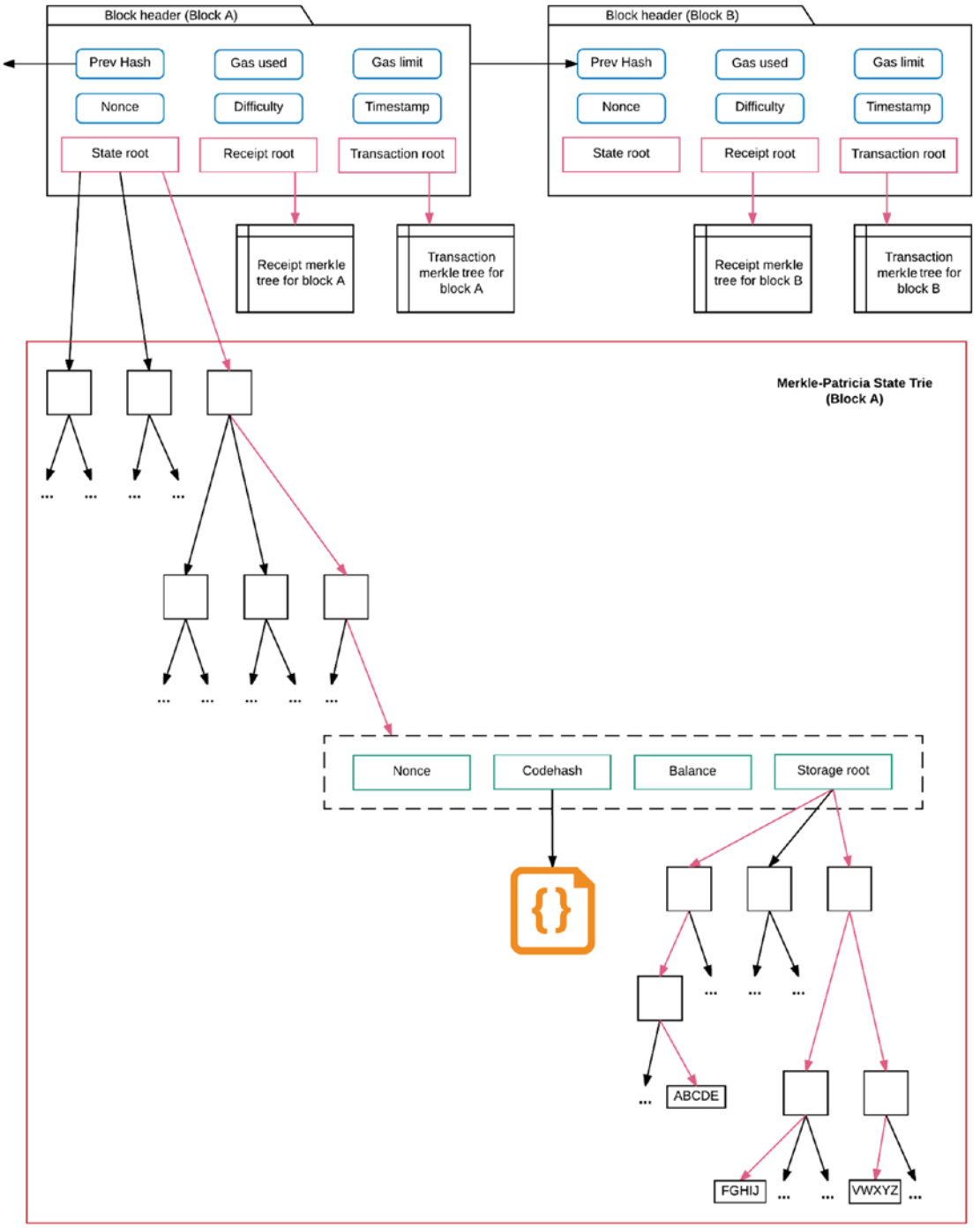


Figure 4-2. *Overview of block headers and Merkle-Patricia trees for Block A and B*

The block header contains a few standard definitions that broadcast the status of the network. Additionally, every block header in Ethereum has three trees for three classes of objects: transactions (function calls), receipts (the results of a function call, recording the effect of each transaction), and state objects. We explore the Merkle-Patricia tree further with the state root, which contains account objects. Binary trees are useful to manage transaction history, but the state has more components and needs to be updated more frequently. The balance of an account and the nonce for the network are often changed and therefore what is needed is a data structure where we can quickly calculate a new tree root after an insert, update, edit, or delete operation without needing to recompute the entire tree. This modified merkle tree allows for rapid queries to questions such as these: Does this account exist? Has this transaction been included in a particular block? What is the current balance of my account? The Merkle-Patricia tree shown in Figure 4-2 is two levels deep and has numerous branches. One of the branches points to a dashed box containing the four components that make up an account. The balance is only relavent for an external account, and similarly, the codehash (which holds executable code) is only applicable to contracts. The storage root actually contains data uploaded by a user to the blockchain or the internal storage space available to a contract that can be updated as that contract is executed.

State, Storage, and Gas

We briefly mentioned that a contract can manipulate its own storage and update the state, so what is a *state*? Recall that in the Bitcoin protocol, data on users and transactions is framed and stored in the context of UTXOs. Ethereum employs a different design strategy of using a state object. Essentially, the state stores a list of accounts where each account has a balance, as well as blockchain-specific data (code and data storage). A transaction is considered valid if the sending account has enough balance to pay for it (avoiding double spending), therefore the sending account is debited and the receiving account is credited with the value. If the receiving account has code associated with it, the code will run when the transaction is received. The execution of a contract or the code associated with an account can have different effects on the state: Internal storage could also be changed, or the code might even create additional transactions to other accounts.

Ethereum makes a distinction between state and history in the network. The state is essentially a snapshot of the current information regarding network state and accounts at a given time. On the other hand, history is a compilation of all the events that have taken place on the blockchain, such as function calls (transactions) and the changes brought about as a result (receipts). Most nodes in the Ethereum network keep a record of the state. More formally, the state is a data structure that contains key value mapping addresses to account objects**.** Each account object contains four values:

- Current nonce value
- Account balance (in Ethers)
- Codehash, which contains code in the case of contracts, but remains empty for external accounts
- Storage root, which is the root of the Merkle-Patricia tree that contains code and data stored on the blockchain

Next, let's talk about gas in Ethereum. Gas is the internal unit for keeping track of execution costs in Ethereum. In other words, it is a microtransaction fee for performing a computation on the blockchain. For a computational platform like Ethereum, this becomes crucial when running code because of the halting problem: One cannot tell whether a program will run indefinitely, or just has a long runtime. Gas puts a limiter on the runtime as the user has to pay for executing step-by-step instructions of a contract. The nature of microtransactions allows steps to be executed very inexpensively, but even those transations will add up for very long runtimes. Once the gas supplied for a contract is exhausted, the user would have to pay for more to continue. Special gas fees are also applied to operations that take up storage.

Operations like storage, memory, and processing all cost gas in the Ethereum network. Let's talk about storage next. In Ethereum, external accounts can store data on the blockchain using contracts. A contract would manage the upload and storage process, but the data types that can be stored currently are very limited. A natural question then becomes: What are the limits on uploading content and information to the Ethereum blockchain? What would prevent the bloating of the blockchain? As it turns out, there are currently two mechanisms in place preventing a data overload:

- Gas limits per block that dictate how much gas can be spent per block on storage and computational operations
- Amount of money a user would have to spend to purchase the gas needed to store data

The second limitation is usually a deterrent for users to store directly on the blockchain, as it becomes much more efficienct and economical to use a third-party decentralized serivce like STORJ (`https://storj.io/`) or IPFS (`https://ipfs.io/`) for the storage and hash the location in Ethereum to include it in a contract. In the future, new distributed storage applications will allow for all sorts of data files to be uploaded and included in contracts on the blockchain. Let's summarize what we have discussed so far: We started with the differences between Bitcoin and Ethereum using accounts, charging gas for operations, storing data directly on the blockchain, allowing executable code on the blockchain, state objects, and Merkle-Patricia trees. Figure 4-3 below a simplified functional overview of the processes occuring in Ethereum.

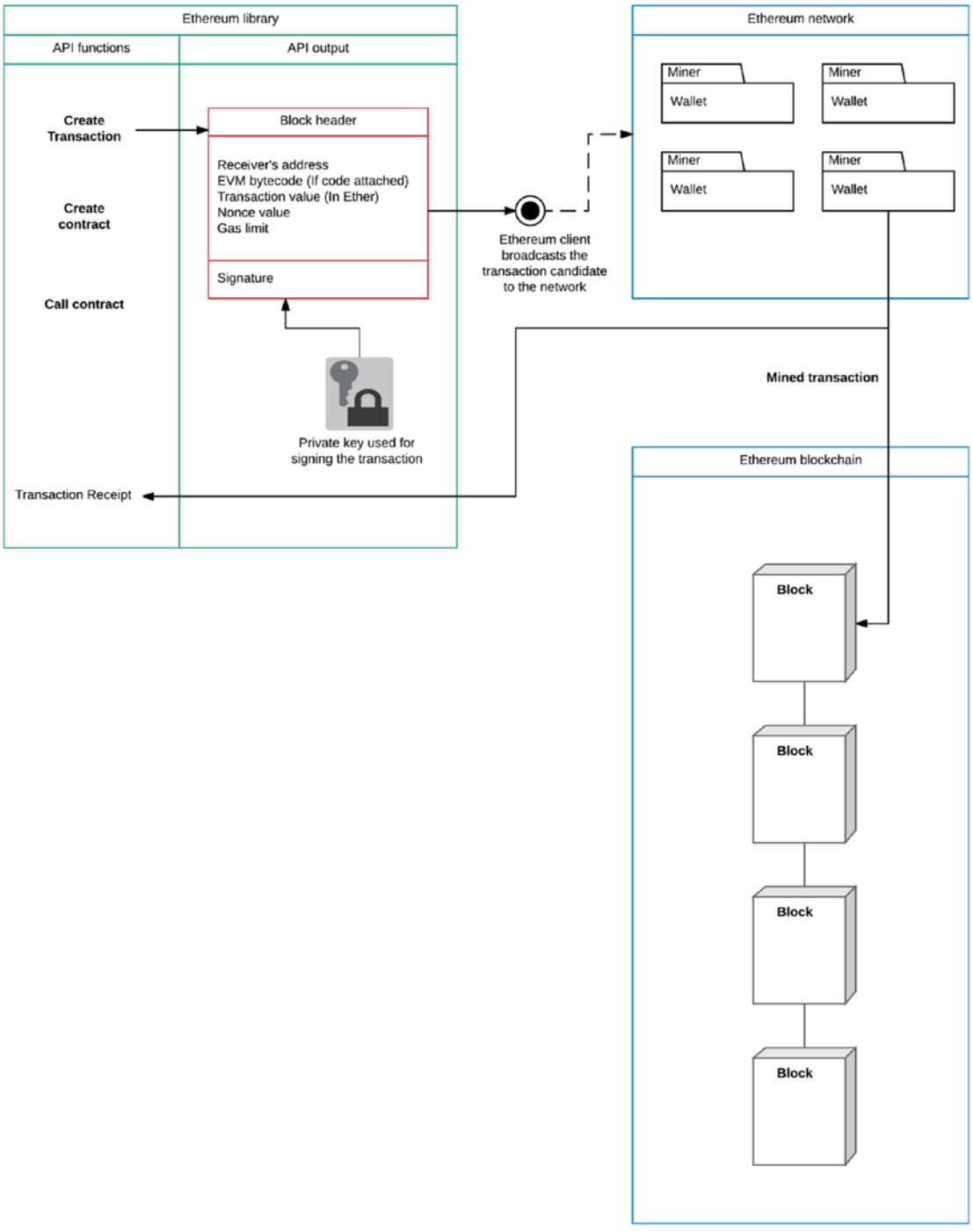


Figure 4-3. *A simplified overview of the Ethereum network*

There are three important Ethereum components to discuss: the API, the network, and the blockchain. The Ethereum JavaScript API (also known as web3.js) provides a large feature set for functionality such as constructing transactions and contracts, referring to functions, and storing receipts. An enhanced wallet client for Ethereum such as Mist (`https://github.com/ethereum/mist`) can take over several of these functions with a GUI. Once a candidate block is constructed, it is broadcast to the network by the Ethereum client. The validators on the network determine if the transactions are valid, and if any code (in the block) associated with a transaction or a contract is valid. Once the validation is complete, the validators execute the associated code and apply it to the current state. The block is broadcast to the network and a miner will forge the block, then the verified block is added to the blockchain. This step also creates transaction receipts for every transaction included in the block. The new block also provides updates to the state objects and relational links for the state from the current block to a new block.

■ **Note** What will prevent the Ethereum network from being bloated by small unused contracts? Currently, there are no mechanisms to control the life span of a contract, however, there are a few proposals in the air about temporary subscription-based contracts. In the future, there might be two different types of contracts, one that has a permanent life span (which is significantly more expensive to create and compute), and the other one that operates until its subscription expires (cheaper and temporary; self-destructs after subscription runs out to prevent cluttering).

Ethereum Virtual Machine

Formally, EVM is the runtime environment for smart contracts in Ethereum. Contracts are written in a higher level language called Solidity and then compiled into bytecode using an interpreter in EVM. This bytecode is then uploaded to the blockchain using an Ethereum client. Contracts live on the blockchain in this executable bytecode form. The EVM is designed to be completely isolated from the environment and the rest of the network. The code running inside the EVM has no access to the network or any other processes; only after being compiled to bytecode do contracts have access to the external world and other contracts.

From an operational standpoint, the EVM behaves as a large decentralized computer with millions of objects (accounts) that have the ability to maintain an internal database, execute code, and talk to each other through message passing. This model is not yet complete, but in Ethereum, this concept is often referred to as the idea of a world computer. Let's return to the topic of code execution and how it is intimately linked to consensus. EVM allows any user on the network to execute arbitrary code in a trustless environment where the outcome is fully deterministic and the execution can be guaranteed. The default execution environment and settings lead to stasis: Nothing happens on the network and the state of everything remains the same. However, as we mentioned before, any user can trigger an action by sending a transaction from an external account. We can have two outcomes here: If the receiver is another external account, then the transaction will transfer some Ether but nothing else happens. However, if the receiver is a contract, then the contract becomes activated and executes the code within. Executing code within the network takes time, and the process is relatively slow and costly. For every step in the instructions, the user is charged gas for execution. When a user initiates an execution through a transaction, they commit an upper limit for the maximum currency that they are willing to pay as gas for that contract or code.

■ **Tip** Ethereum has recently begun the process of migrating over to a just-In-time virtual machine (VM), which offers some optimizations in gas usage and performance.

What does it mean for the outcome of EVM to be deterministic? It is essential for each node to reach the identical final state given the same input for a contract method. Otherwise, each node that executes the contract code to validate the transaction would end with different results and no consensus would be possible. This is the deterministic nature of EVM that allows every node to reach consensus on execution of a contract and the same final state of accounts. The nodes executing a contract are similar to cogs synchronized to move inside of a clock, as they work in a harmonious fashion and reach the matching final state. A contract can also refer to other contracts, but it cannot directly access the internal storage of another contract. Every contract runs in a dedicated and private instance of the EVM where it only has access to some input data, its internal storage, the code of other contracts on the blockchain, and various blockchain parameters such as recent block hashes.

Every full node on the network executes the contract code simultaneously for each transaction. When a node is validating a block, transactions are executed sequentially, in the order specified by the block. This is necessary because a block might contain multiple transactions that call the same contract, and the current state of a contract might depend on state modified by previous references during the code execution. Executing contract code is relatively expensive, so when nodes receive a block, they only do a basic check on the transactions: Does the sending account have enough Ether to pay for gas? Does the transaction have a valid signature? Then, mining nodes perform the relatively expensive task of executing the transaction, including it in a block, and collecting the transaction fee as a reward. When a full node recieves a block, it executes the transactions in the block to independently verify the security and integrity of the transactions to be included in the blockchain. Let's look at the EVM visually in Figure 4-4.

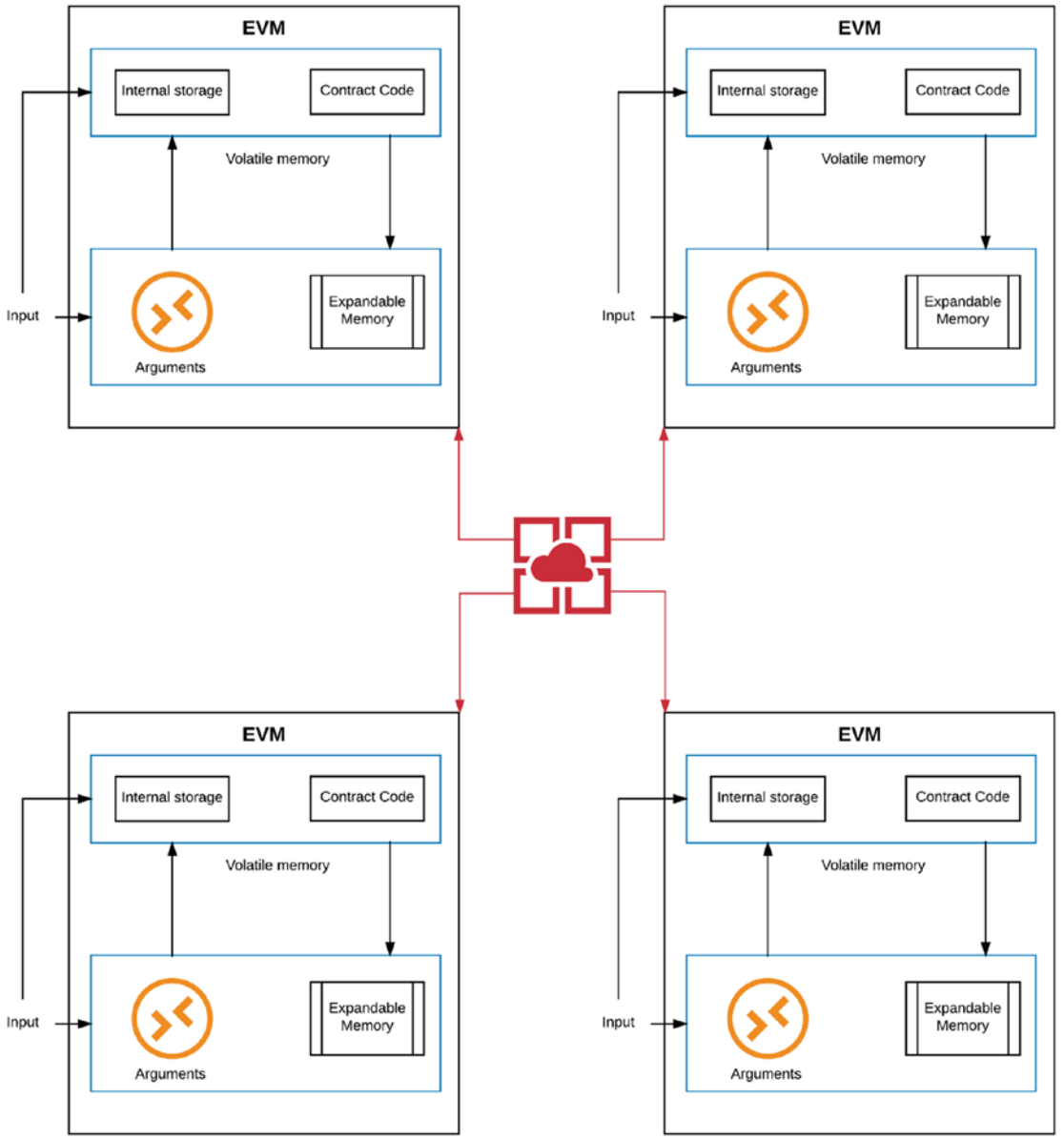


Figure 4-4. *Four instances of Ethereum Virtual Machines (EVMs) running on four different nodes*

The four EVMs are synchronously executing a contract's instructions and will arrive at the identical account state once the execution has been completed. This is due to the deterministic nature of the EVM, which allows the contract to reach consensus across the network at every step of instructions. The EVM has a very straightforward rationale: It has a single run loop that will attempt to execute the instruction one step at a time. Within this loop, the gas is calculated for each instruction and the allocated memory is expanded if necessary. The loop will continue until the EVM either recieves an exit code indicating successful execution or throws an exception such as out of gas.

Solidity Programming Language

Solidity is a higher level, object-oriented programming language for writing smart contracts in Etheruem. Any code written in Solidity can be executed on the EVM after being compiled into bytecode, which is an instruction set for the EVM. How does the bytecode encode references to other functions and contracts that are called during execution? This is done using an application binary interface (ABI). In general, an ABI is the interface between two program modules: machine-level instructions and a human-readable higher level programming language. Let's break down this answer into three components:

- *Contract*: A contract is simply higher level code defined in a formal language such as Solidity.
- *Compiled contract*: The contract is converted to bytecode to be executed on the EVM, adhering to the compiler specification. Note that function names and input parameters get hashed and obfuscated during compilation. Therefore, for another account to call a function, it must have access to the given function name and arguments, and we need another layer that interfaces encoding into and out of the bytecode.
- *ABI*: An ABI is a list of the contract's function definition and arguments in JavaScript Object Notation (JSON) format. The function definitions and input arguments are hashed into the ABI. This is included in the data of a transaction and interpreted by the EVM at the target account. An ABI is necessary so that you can specify which function in the contract to invoke, as well as get a guarantee that the function will return data in the format you are expecting.

Solidity (see https://solidity.readthedocs.io/en/develop/) has a new plug-in for Visual Studio (see https://marketplace.visualstudio.com/items?itemName=ConsenSys.Solidity) to help write smart contracts in a powerful IDE and deploy them to the Ethereum network. Our discussion of Solidity here is limited to covering the fundamentals, such as storing variables and creating a simple contract, so let's get started.

```
/* defining a contract */
contract ExampleStorage {

    uint storedNumber; //unsigned integer (uint) used to declare a state variable

/* Function set can modify the value of the state variable */
    function set(uint x) {
        storedNumber = x;
    }

/* Function get can retrieve the value of state variable  */
    function get() constant returns (uint retVal) {
        return storedData;
    }
}
```

This storage contract allows a user to store an integer as a state variable `storedNumber` and then modify or retrieve its value using the `get()` and `set()` functions. Solidity also offers several advanced features available in modern programming languages such as inheritance (for contracts), function overloading, and class interfaces. Next, let's look at a more complex example of a contract. This time we create a simple bank contract using Solidity:

```
// This bank contract allows deposits, withdrawals, and checking the balance

// 'contract' is a keyword to declare class similar to any other OOP
contract SimpleBank {

// 'mapping' is a dictionary that maps address objects to balances
// 'private' means that other contracts can't directly query balances
    mapping (address => uint) private balances;

// 'public' makes externally readable by users or contracts on the blockchain
    address public owner;

// Events trigger messages throughout the Ethereum network
    event LogDepositMade(address accountAddress, uint amount);

// Constructor
    function SimpleBank() {

        // msg provides details about the message that's sent to the contract
        // msg.sender is the address of contract creator
        owner = msg.sender;
    }

    // Deposit Ether into the bank
    // Returns the balance of the user after a deposit is made
    function deposit() public returns (uint) {
// Add the value being deposited to the account balance
        balances[msg.sender] += msg.value;

// Log the deposit that was just made
        LogDepositMade(msg.sender, msg.value);

// Return the balance after the deposit
        return balances[msg.sender];
    }

// Withdraw Ether from bank
// withdrawAmount is the amount you want to withdraw
// Returns the balance remaining for the user
    function withdraw(uint withdrawAmount) public returns (uint remainingBal) {
```

```
/* If the account balance is greater than amount requested for withdrawal, subtract it from
the balance */
        if(balances[msg.sender] >= withdrawAmount) {
            balances[msg.sender] -= withdrawAmount;

// Increment the balance back to the original account on fail
            if (!msg.sender.send(withdrawAmount)) {
                balances[msg.sender] += withdrawAmount;
            }
        }

// Return the remaining balance after withdrawal
        return balances[msg.sender];
    }

// Return the balance of the user
// 'constant' prevents function from editing state variables;
    function balance() constant returns (uint) {
        return balances[msg.sender];
    }

}
```

Although this contract has plenty of moving parts, it has a straightforward schematic: We start by declaring state variables and here we used an advanced data type called a mapping. Then, we declare an address variable used throughout the contract and an event logger. The constructor prepares the owner object to be usable and we attach the owner object to recieve messages in the form of return types from functions. There are three functions that follow the constructor that allow for the basic functions of a bank. The deposit function adds the argument amount to the balance. The withdrawal function checks whether the requested amount is lower than the balance available for an account. If this is the case, the withdrawal is confirmed and the argument amount is subtracted from the balance. If there is not enough balance, the amount that was supposed to be withdrawn is added back to the account and the final balance is returned to the user. Finally, the last function allows us to return the balance of an account at a given time as requested by the contract.

World Computer

The Ethereum project has a grand vision of becoming a shared world computer with millions of accounts, powered by the blockchain, which becomes a back-end for smart-logging of communications. Contracts provide the decentralized logic to be executed and EVMs are the execution platform. Computation and processing are not enough, though; a computer must also be able to store information and allow for a mechanism for applications to communicate among each other. This world computer, depicted in Figure 4-5, would operate in an Internet 3.0 era, where servers are no longer needed due to the decentralized nature of information flow. In this ambitious endeavour, Ethereum is only one third of the project, so let's introduce the other two components:

- *Whisper*: A message-passing protocol that allows decentralized applications and accounts on the blockchain to communicate with each other. This is different from traditional message passing protocols where applications execute on the same machine; here, the decentralized apps can execute on any node in the blockchain.

- *Swarm*: A decentralized data storage and distribution resource available to the Ethereum blockchain. Swarm is a peer-to-peer data sharing network where files are addressed by the hash value of their content. This resource is very similar to BitTorrent, where data can be fetched from multiple nodes (called peers) that host small pieces of a file(s), and they are put together by the receiving party. Swarm's most powerful feature is that if even a single node hosts a piece of data, it can be accessed from anywhere within the network. Currently, Swarm is in the early stages of development and doesn't specify one service (Storj, IPFS) that will provide decentralized storage; however, Swarm does have the tools to handle storage and hashed references to the data stored off-blockchain. Using Swarm makes it possible to distribute data across the network to replicate redundancy in a decentralized fashion without having to host any kind of server. Multiple nodes in the network can be incentivized to replicate and store the data, much like a RAID configuration, eliminating the need for hosting servers in the network.
- *Contracts*: These are the final component of the world computer that we mentioend previously. They allow programmatic access to the blockchain and provide the logical framework to power applications that will eventually run on the world computer.

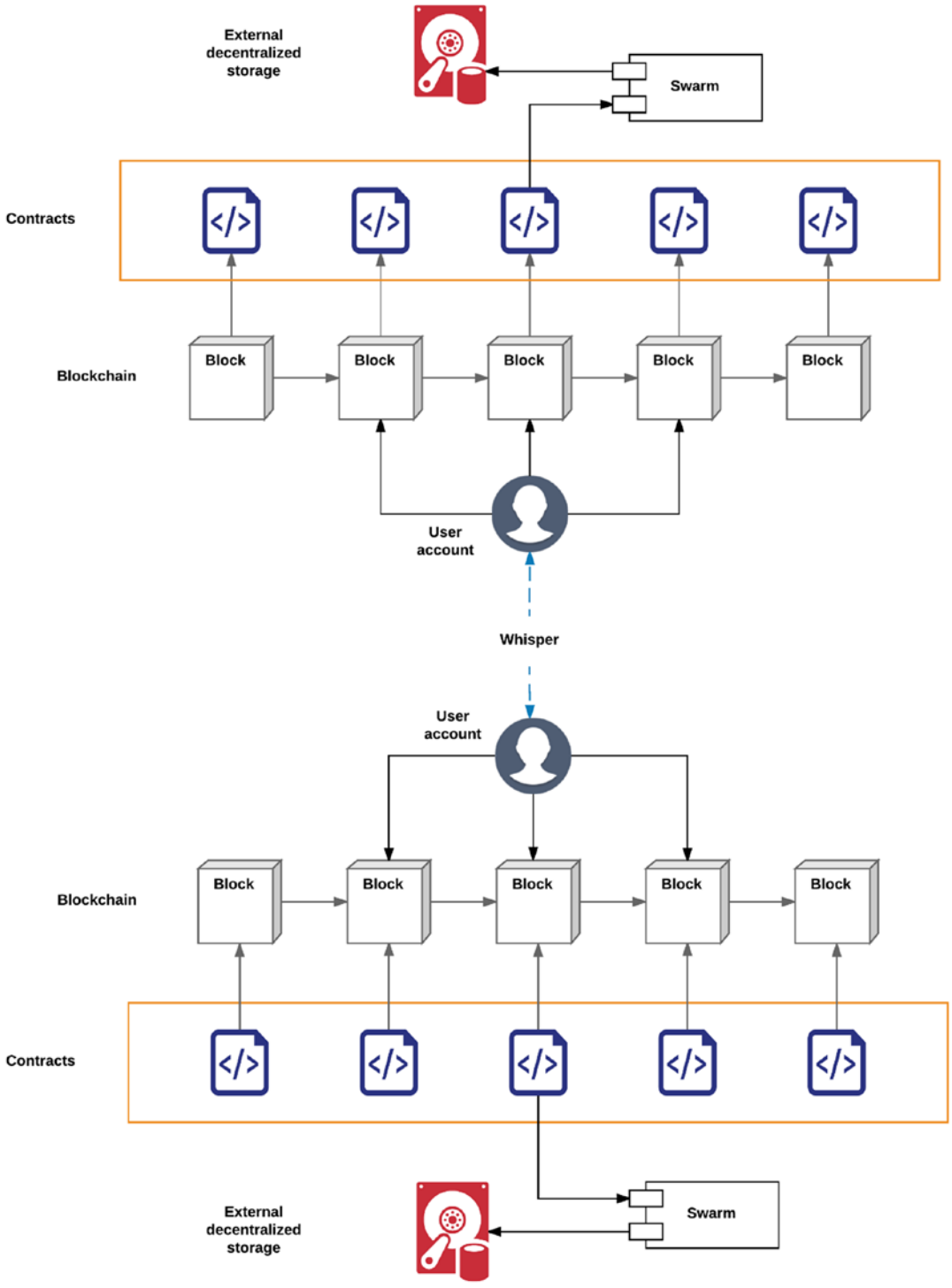


Figure 4-5. *A layered approach to the world computer model in Ethereum*

User accounts (or simply users) on the world computer are fundamental entities in the world computer, so users are the first layer. The second layer is the blockchain, which serves as a communication serial bus between the different components in the network. The third layer is the logical framework called smart contracts that reside on the blockchain, and provide the computational capabilities to the world computer. Some of these contracts might require external storage for output and use Swarm to coordinate storage, which is the fourth layer. Finally, looking back at the first layer, we have the message passing protocol called Whisper to facilitate user-to-user or application-to-application communication.

More than a philosophical vision or a technological blueprint, the concept of a world computer and Internet 3.0 have some far-reaching implications for how content is controlled and distributed across the Web. Taylor Gerring from Ethereum spoke very eloquently about building this dream:

> *As economics of the Ethereum ecosystem mature such that open contracts for lowest-rate storage develop, a free market of content hosting could evolve. Given the nature and dynamics of P2P applications, popular content will readily scale as the swarm shares, rather than suffering from the buckling load of siloed servers. The net result is that popular content is delivered faster, not slower.*
>
> *This metamorphosis will offer developers an opportunity to build the next-generation of decentralized, private, secure, censorship-resistant platforms that return control to creators and consumers of the next best idea. Anyone with a dream is free to build on this new class of next-generation decentralized web services without owning a credit card or signing up for any accounts.*
>
> *Although we are not told to or expected to, we have an imperative to cherish and improve the very shared resources that some wish to disturb, manipulate, and control. Just as no single person fully understands the emerging internet collective intelligence, we should not expect any single entity to fully understand or maintain perfectly aligned motives. Rather, we should rely on the internet to solve the problems of the internet.*

Blockchain-as-a-Service

Microsoft recently announced a partnership with the Ethereum Foundation to launch a blockchain-based service on their cloud platform Azure. This Infrastructure-as-a-Service approach to offering fast and easy implementations of blockchain will allow developers to experiment with new features and deploy DApps at reduced costs. Marley Grey from the Azure Blockchain Engineering team described how Blockchain-as-a-Serivce (BaaS) will foster an ecosystem of DApps:

> *Microsoft and ConsenSys are partnering to offer Ethereum Blockchain as a Service (E-BaaS) on Microsoft Azure so Enterprise clients and developers can have a single-click cloud-based blockchain developer environment. The initial offering contains two tools that allow for rapid development of SmartContract based applications: Ether.Camp, an integrated developer environment, and BlockApps, a private, semiprivate Ethereum blockchain environment, can deploy into the public Ethereum environment.*
>
> *"Ethereum Blockchain as a Service" provided by Microsoft Azure and ConsenSys allows for financial services customers and partners to play, learn, and fail fast at a low cost in a ready-made dev/test/production environment. It will allow them to create private, public and consortium based Blockchain environments using industry leading frameworks very quickly, distributing their Blockchain products with Azure's World Wide distributed*

> *(private) platform. That makes Azure a great Dev/Test/Production Environment for Blockchain applications. Surrounding capabilities like Cortana Analytics (machine learning), Power BI, Azure Active Directory, O365 and CRMOL can be integrated into apps launching a new generation of decentralized cross-platform applications.*

This initial update on BaaS was provided at the end of 2015 and currently a whole ecosystem of Blockchain Labs is flourishing within the Azure DevTest community. The DevTest Labs allow users and developers to explore and test a template designed for a specific use case. In addition, the platform began with Ethereum blockchains but recently more startups have started to build on Azure, offering new services such as Emercoin, which offered an SSH service, and PokiDot, with its health-care-oriented blockchain called Dokchain. Over time, more startups have begun using Azure as the standard to run a blockchain and build applications on top. With the integration of intelligent services such as Cortana, it might become easier to develop oracles that can sign incoming data from external streams (e.g., IoT devices) and provide a level of integrity.

■ **Note** Two recent developments from Microsoft in the BaaS space are noteworthy here. The first is introduction of Cryptlets, a secure middleware to interface with external events for building enterprise smart contracts. The second development is the Coco framework. An open-source system for building a high throughput network on top of a blockchain, where nodes and actors are explicitly declared and controlled. By design, Coco is compatible with any ledger protocol, and would allow enterprises to build production-ready blockchain networks.

Decentralized Applications

We alluded to DApps in our discussion of Whisper, but we will talk about them in more depth here. A DApp is a serverless application that runs on the Ethereum stack and interfaces with the end user via an HTML/JavaScript front end that can make calls to the back-end stack. Clasically, a mobile or web app has a back end running on centralized dedicated servers; however, a DApp has its back-end code running on a decentralized peer-to-peer network. The structure of a DApp is shown in Figure 4-6.

Figure 4-6. *Structure of a DApp*

The user interace is often written in HTML or JavaScript and it is the only component loaded on a user device. The interface makes back-end calls to the blockchain to execute a particular contract and also to the back-end resources such as Swarm or Whisper if external storage is needed or when the application needs to communicate with other apps.

If a traditional app is made up of a front end and a server running the back end, then a DApp running on the Ethereum stack would be made up from a front end and contracts running on the blockchain. DApps usually have their own set of associated contracts on the blockchain that are used to encode business logic and allow persistent storage of their consensus-critical state. Recall that all code on the Ethereum stack runs within an EVM that keeps track of step-by-step operations and charges gas to the owner of a contract. This prevents DApp developers from running too many operations on the blockchain or bloating it by storing data directly on the blockchain.

■ **Note** To briefly review, the Ethereum stack is made of three components: the blockchain, Whisper, and Swarm. The front-end interface makes calls to the blockchain to specific contracts running the DApp based on user actions.

How does the back end of a DApp pull static content for the front end such as JavaScript from the Ethereum stack to static content and receive the updated global state from the blockchain? Let's look at an example using IPFS as storage to understand these back-end calls, as depicted in Figure 4-7.

- The back-end code is essentially a contract that executes on the blockchain given the appropriate amount of resources.
- Some applications need to use a persistent database to host static content used in the app. We can rely on IPFS that stores static files, hosted throughout the network on several nodes.
- Hashes from IPFS are delivered to the DApp and the contract's execution updates the global state, which is deliverd to the DApp from the Ethereum stack.

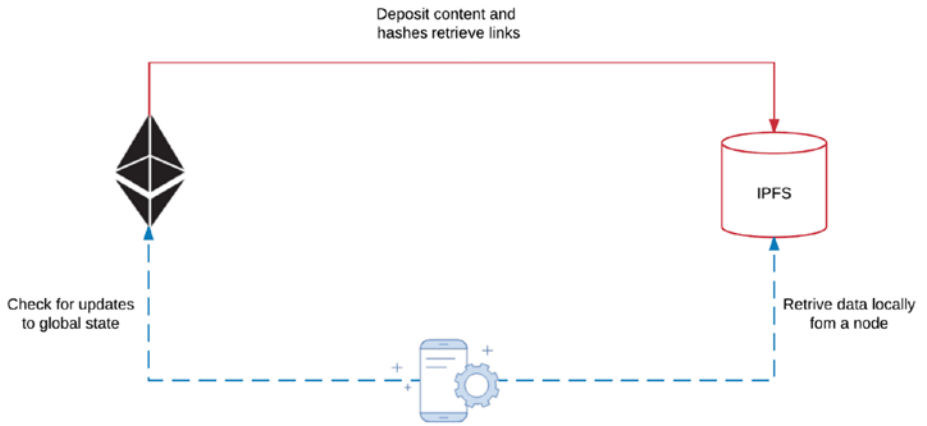


Figure 4-7. *A simple schematic of back-end calls made by a DApp*

The blockchain can deposit content to an IPFS-like system on one of the nodes and the hashes can be made available to the app for retrieval when necessary. The app can request updates from the blockchain on the global state as it affects the app running on a device. Finally, as needed, the app can retrieve and download full content from the decentralized storage to the user device. Splitting up roles in this manner allows for more innovative user interfaces, as a developer can switch it out, not having to change the back end at all.

Geth and Mist

There are two more tools we need to discuss briefly that play a role in DApp develpoment. Geth is the command-line interface (written in Go-lang) for running a full node on the Ethereum network. Using Geth, you can interact with the Ethereum network and perform tasks such as:

- Mine Ether on the network
- Transfer funds between addresses
- Create contracts and send transactions
- Use the DApps API

Geth comes with two interfaces that are used in development: The JavaScript console with web3.js library, and the JSON-RPC server. Let's talk about both technologies briefly. Geth can be launched with an interactive console that provides the JavaScript runtime environment for you to interact with a node. This runtime environment includes the web3 library, which can construct contracts and transactions to be propagated to the node. The JSON-RPC server is a remote procedure call (RPC) protocol that facilitates data exchange between the node and its clients (JSON is a data-exchange format that nodes use to communicate with clients). More precisely, RPC is a collection of methods and rules that define how data (commands and output) can be transferred between a node and a client. The JavaScript API uses the web3.js library to offer a convenient interface for using the RPC methods.

■ **Tip** For most Ethereum applications today, Geth is a prerequisite for installation as a command-line tool. Often, during the installation, Geth is provided as an add-on so that a user doesn't have to download and install it separately.

The second tool is called the Mist DApp browser. In early discussions, Mist was conceptualized to be a stand-alone app-store-type browser for DApps, but that vision has evolved. Mist is still in heavy development; however, the release plan is to bundle Mist together with the Ethereum wallet to make a powerful tool. In future releases, the wallet will just be a single DApp running on the Mist browser. Soon the browser will be able to open any DApp available and the wallet will just be one app among them.

Eventually, the most powerful entitites on the Ethereum network using Geth and Mist will be decentralized autonomous organizations (DAOs), which are essentially automated companies powered by smart contracts that run on the Ethereum network. We end our journey exploring Ethereum here and pick up our discussion of DAOs in the next chapter.

Summary

In this chapter, we introduced Ethereum, one of the biggest alternate currencies competing against Bitcoin. In recent years, it has gained a very serious focus from developers and investors. We began our discussion here with a broad overview of what Ethereum is in comparison to Bitcoin. We talked about accounts and function calls as being foundational to Ethereum. We then provided some more depth to the ideas of accounts as entities on the blockchain. After that, we discuss the use of gas on Ethereum for smart contract execution, how internal storage is adapted to work with Merkel-Patricia trees, and the concept of internal state for an account. After that, we talked about EVM and how smart contracts are executed on the

blockchain. Then, we discussed a model for writing smart contracts using Solidity and applying a plug-in for Visual Studio to rapidly prototype smart contracts. Finally, we talked about the world computer model as it applies to Ethereum components such as IPFS and Whisper. We ended the discussion with a short description of all the components in the world computer.

References

The main reference material used to prepare this chapter was Ethereum Homestead developer documentation and Solidity documentation. A detailed list of references is given at the end of the book.

CHAPTER 5

Decentralized Organizations

Bitcoin can be thought of as the first prototypical decentralized autonomous organization (DAO). It created a network-based ecosystem of participants who contributed computational power toward a singular goal. In Bitcoin, the distributed protocol providing a financial service and rewarding miners became a rudimentary decentralized organization. In this chapter, we talk about more complex and full DAOs made in Aragon. Aragon (https://aragon.one/) is a decentralized application (DApp) that lets anyone create and manage different kinds of organizations (nongovernmental organizations [NGOs], nonprofits, foundations) on the Ethereum blockchain. Creating a DAO requires numerous steps and originally it was more difficult to implement in Ethereum. However, Aragon implements all the basic features of an organization in a base template that is deployed whenever a user instantiates a company. Most of the traditional features such as a cap table, voting, fundraising, and accounting are offered in Aragon as a decentralized counterpart to run on the blockchain. In addition, an Aragon company can be customized to a very granular extent and extended using new modules that can be added to a company's existing smart contracts. Aragon enables different organizations to be built on the blockchain, and one interesting use case integrates identity using a two-way verification scheme with Keybase. We talk about how the Keybase to Aragon peg functions to provide identity services in the context of a decentralized system. We also briefly go over the Aragon kernel, which is essentially a task manager with subroutines that ensure smooth communication within an organization, among its members, and in the underlying blockchain.

We begin our discussion with the Aragon kernel and it's main functions. Then, we take two approaches in defining a DAO: A definition in terms of consensus between members, and a definition focusing on features given by Ralph Merkel. Aragon-core has adopted the latter definition, and extended it to "organs", so we go through all the organs available to Aragon-core. The heart of this chapter is a set of visual tutorials split into three topics. The first topic introduces users to the basics of an Aragon company and how to set up a MetaMask wallet. This topic ends with users having set up a default company and a functional wallet. The second topic dives into understanding the daily operations of your newly created company. This covers stocks and stock classes available in Aragon, user permissions, and tokens. A variety of operations are demonstrated here, such as adding new stock classes, issuing shares, assigning stocks to a new hire, assigning a role to a new employee, and transferring existing tokens within the company. The third topic focuses on more advanced topics in Aragon. Here, we discuss fundraising on the Ethereum blockchain using Aragon and the different types of rounds implemented in Aragon. Finally, we conclude the tutorial and this chapter with a discussion of bylaws and how to edit the default company template deployed by Aragon.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_5

Aragon Kernel

The kernel in Aragon serves as a switchboard directing requests and message passing to the various subroutines (also called organs). In production mode, the kernel will be concurrently interfacing with hundreds of Aragon organizations or users on the blockchain. Here, our focus is limited to the kernel interacting with just one company: a DAO. What is a DAO? A DAO is a blockhcain entity built on a consensus of decisions by its members. A DAO brings the concept of consensus inherent to the blockchain to include decisions and choices made by members of a DAO that have a stake in it (usually in the form of DAO tokens). As such, a DAO is built with automation at its core and members at the periphery that rely on a majority-based consensus to make decisions for the organization. Although this definition of a DAO is a general descriptor, it is incomplete. Perhaps a better approach to defining a DAO would be through the functions it performs. The concept for a DAO was originally derived from Bitcoin, which might be viewed as the first prototype for a DAO.

■ **Note** The simplest functional unit on an Aragon network is a DAO and therefore the majority of our discussion is focused around a minimal use case. Aragon provides a base template that can be used to set up a DAO and you can modify this template to set up a custom set of rules. Other types of organizations (e.g., NGOs or nonprofits) are built on the base template with significant modifications that allow a new modus operandi. Additionally, throughout the remainder of this chapter, the terms DAO, company, and organization are used interchangeably to represent the same concept.

Ralph Merkle talks about a DAO as an entity that owns internal property that has value, can update its internal state, is responsive to the members, and runs smart contracts. These are some of the most basic functions that any DAO should be able to perform, but currently there are no compliance standards (like the ERC20 for tokens) for DAOs. The kernel uses a set of organs to carry out most of its daily activities. Let's go through the organs provided by default in Aragon:

- *Meta organ*: A self-aware (in terms of internal state) and self-actionable organ that is responsible for updating the DAO internally in response to actions by the members or externally. This organ also maintains a global registry of organs operating within the kernel.
- *Dispatch organ*: A validator organ that determines whether a requested action or a transaction can be carried out on behalf of the requestor. This can be done through an oracle, or programmed logic that filters requests based on specific criteria. The outcome of the dispatch organ is a simple pass or fail. If an action fails, it will not be carried out. However, if an action or transaction passes, it will be dispatched to a more appropriate organ for processing or execution. The dispatch organ also maintains a list of priorities for each request to be triaged and directed appropriately.
- *Vault organ*: The vault organ serves as a collective wallet for the DAO. It stores the funds owned by the DAO as tokens and approves any spending requests.
- *Token organ*: The token organ specifically deals with governance tokens allocated to the members of DAO. This organ also contains the operational logic for adding new types of tokens or replacing and removing older tokens.
- *Applications organ*: This is the smart contract collective that operates at the core of a DAO. The applications running in this organ are sandboxed from the remainder of the organization, but most of the business logic resides in this organ. Aragon provides a basic set of Ethereum contracts responsible for default actions, but this organ is extensible. New applications or modules can be added to the organization to increase functionality to satisfy specific use cases.

Identity Management

The concept of identity is a conundrum for cryptotechnologies because it requires a certain level of trust to be inherent in the network architecture. Most generalized approaches toward integrating identity in a consensus-based system involve some variation of cryptographic proofs and signatures. In Aragon, an external service called Keybase along with a registry contract are used to establish a "trustless" two-way verification scheme. The logic behind this scheme is very straightforward: Establish that a particular address belongs to you, and then verify that your username owns the address. How do these two statements reconcile in a functional setting? Let's introduce the two components that make a two-way bridge possible. In simple terms, Keybase is a public–private key pair management service with OAuth integrations that allow a user to authenticate and verify accounts. Keybase can serve as a centralized hub for a user to link and verify external accounts. For instance, after creating an account on Keybase, the public profile of a user can display all the linked social media accounts establishing their legitimacy and connection. The second component is a registry contract that provides an account-to-address association mechanism from within the Aragon network. The two-way bridge forms from using the address (linked to an account on the network) to cryptographically sign a file that can then be hosted on Keybase (linked to user identity through various social media integrations).

■ **Note** It is crucial to keep in mind that in Aragon, identity is an opt-in feature. The use cases developed for Aragon range from providing complete anonymity, as in the case of a DAO, to full usernames integrating Keybase for a next-generation blockchain company.

So how would users make their identity available to the two-way verification scheme? The workflow in Keybase Registry 2.0 has been simplified to one simple step: Upload the signed proof to the Keybase Filesytem (KBFS). This introduces a few new components.

- *Keybase Filesystem*: KBFS is a cryptographically secure local and cloud storage directory. KBFS only allows the owner of the directory to add new files, and these files are made available publicly. In KBFS, every file added to the directory is automatically signed by the owner's private key and the signature can be verified by anyone through Keybase.
- *Signed proof*: Aragon uses a standard proof of identity signed by the user uploaded to the KBFS public directory. There are four components of this proof: the username on Keybase, the Ethereum account address owned by the user, a string or comment by the user, and finally the public key of the signature. Anyone who wishes to verify an identity can obtain this proof file and use a function such as `getUsername(args)` to perform a username reverse lookup given the account address in the proof.
- *Oraclize*: The reverse lookup is an on-chain verification process performed through the registry contract within the Aragon network. For Keybase lookup, a data carrier service called oraclize is used. Oraclize requests the signature for a given Keybase username and creates a mapping of the username to an Ethereum account address. The signature on this proof file should validate with the user's public key on the network. Oraclize performs an on-chain check to verify the proof signature and ensure that the mapping is legitimate. The party initiating this verification on behalf of another user also has to pay for the steps, including oraclize reference calls.
- *Keybase Resolver*: The Ethereum Name Service (ENS) allows us to resolve human-readable links into Ethereum addresses. Recently, Aragon has begun testing a Keybase resolver that will map callbacks to usernames into addresses such that `john.keybase.eth -> 0x99...cx88`. This simplifies the reverse lookups and username references throughout the network.

DAO/Company Walkthrough

In this section, we go through the process of creating a DAO and becoming familiar with the major DAO operations in Aragon. For the sake of simplicity, we will only have two entities participating in this DAO, an executive and an employee. This walkthrough is split into three topics:

- *Setting up a DAO*: We introduce the participants of this DAO, show how to create a wallet, and get familiar with the interface.
- *Creating stocks and assigning shares*: After our DAO has been set up, we create new stock classes, issue new stock, and assign shares to the other entity.
- *Fundraising and editing bylaws*: After shares have been assigned, we look at how fundraising works in Aragon. A DAO can raise funds from a specific investor in return for stocks, or publicly offer stock to raise a round of funding. We review the fundraising process, and more important, the bylaws that govern the DAO.

Setting Up a DAO

Download the Aragon client from `https://aragon.one/`. This client has two main components: Aragon Core and MetaMask. In each client, Aragon Core functions as an administrative dashboard for the DAO. The main purpose of this component is directing access control of all entities participating in a DAO. Depending on your role and if you hold any stock, you might have access to the advanced features. We see later how this access to features can change by editing the bylaws. The second component is MetaMask, a digital wallet designed as a plug-in for Google Chrome to make Ethereum accessible to pragmatic daily users. MetaMask was chosen for Aragon to drive adoption by bringing an already familiar interface to the user. The opening screen, shown in Figure 5-1, provides a brief introduction to Aragon, along with the wallet setup.

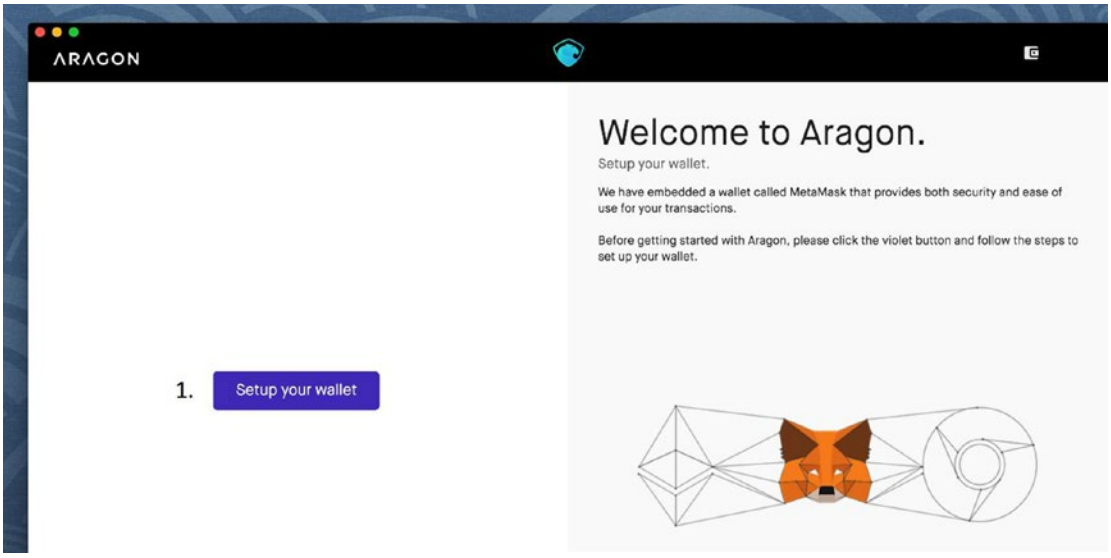

ARAGON
Welcome to Aragon.
Setup your wallet.
We have embedded a wallet called MetaMask that provides both security and ease of use for your transactions.
Before getting started with Aragon, please click the violet button and follow the steps to set up your wallet.
1.
Setup your wallet

Figure 5-1. *The Aragon setup screen. A brief introduction to Aragon is provided here, along with a button to create a MetaMask wallet. On the top right of the screen there is an icon to access the wallet.*

MetaMask first asks you to create a password for your wallet, as shown in Figure 5-2. This will be your access key. After entering the password, you will be provided with a passphrase to recover your wallet in case you lose your password.

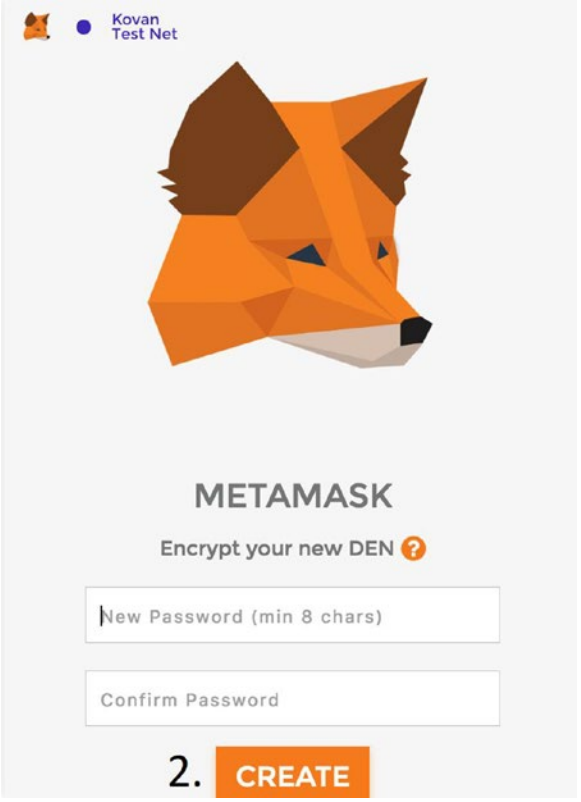


Figure 5-2. *Creating a password for your MetaMask wallet. Every major step in Aragon is followed by a series of confirmations that broadcast new changes to the network in the form of transactions.*

■ **Note** Aragon is still connected to a testnet to illustrate the concept. As a result, for some steps, you might need to verify and confirm transactions multiple times to ensure the initial company creation steps. Some of the bugs are known and under active development in the beta code.

After creating a wallet password and confirming the transactions, you should see a confirmation screen and the state of your current wallet (see Figure 5-3). Here is a quick note about Aragon's alpha releases: Currently the Aragon client does not operate on the Aragon network. It is connected to an Ethereum testnet (Kovan Test Net in this case, as shown in Figure 5-3), which serves as a testing ground for Aragon. On a testnet, currency holds no value, so users can test the product and report any serious bugs, assisting in product development. A user can request test ETH to be sent to their wallet and experiment with Aragon without any consequences. That's why when the wallet is created, it already comes loaded with a balance.

Figure 5-3. *Wallet created using MetaMask. Notice the wallet address shown below along with the balance it has on the testnet.*

The screen after wallet creation brings you to Aragon Core, and there are two options available: Create a New Organization and Join an Existing Organization, shown in Figure 5-4. From here on, the walkthrough focuses on the first option, and Aragon is presented from the point of view of an executive. However, we briefly touch on how users can join a DAO and the level of access available to a regular employee.

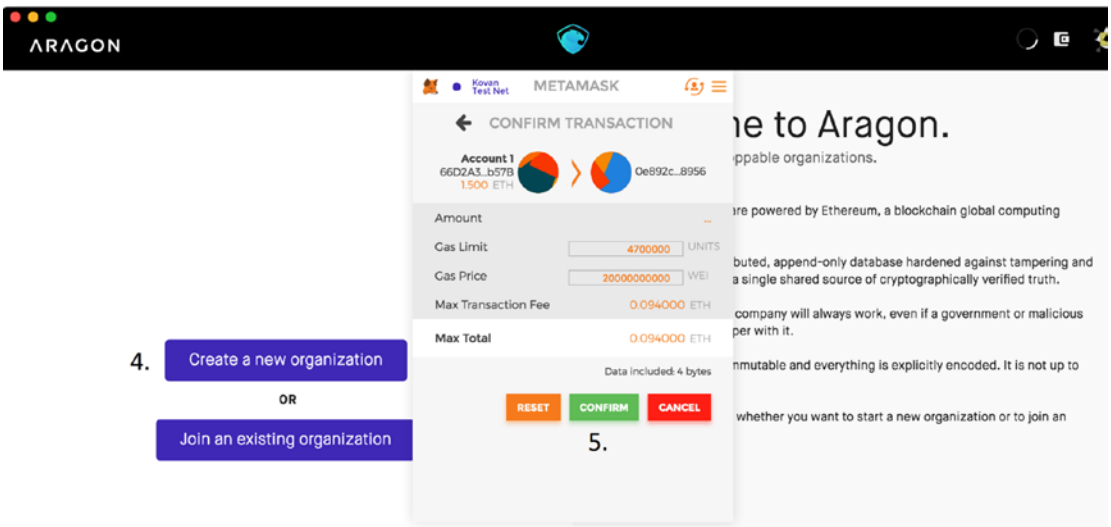


Figure 5-4. *Aragon Core Welcome screen*

The two options provided to a user are not limited to just DAOs; rather, any type of organization can be built on Aragon, including NGOs and nonprofits. Traditional companies would take a new decentralized outlook on the Aragon blockchain, but ultimately it is up to the companies to decide which components can be decentralized. We will select Create a New Organization here. Creating a new company costs Ether, paid with the wallet we created in the previous step. Then, the transactions are broadcast to the blockchain by confirming them. Aragon uses a general template to deploy a company on the blockchain, but the founders can edit this template by changing bylaws. A circular progress indicator in the top right corner shows the transactions being verified for company deployment.

After the company has been deployed, the next screen is the Aragon Core dashboard. This is where a user (executive or employee) handles the daily operations of a DAO, with functionss to participate in voting, assign shares, and more. The dashboard displays the welcome screen by default to the user. To navigate this dashboard, let's begin by looking at your profile under Settings (Figure 5-5).

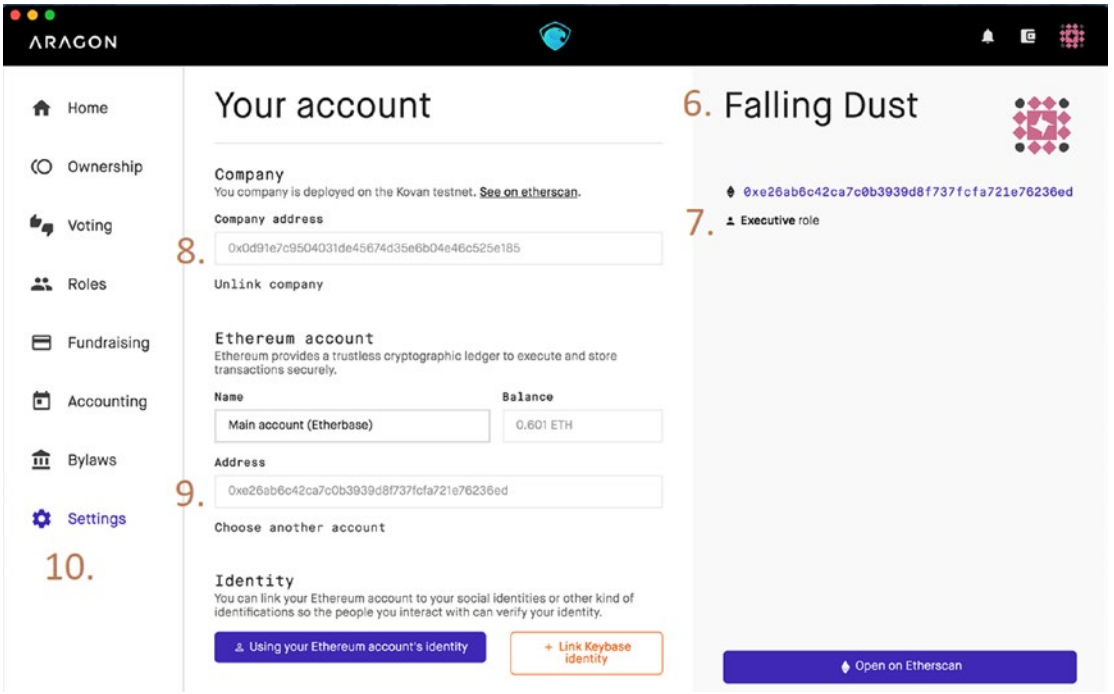


Figure 5-5. *Settings in Aragon Core dashboard*

Each account is assigned an address and a username on the network. This address is also associated with the funds available to the account through the wallet. Under the address is the role of this user. Next, the Settings screen shows some information about the company we created in earlier steps. The address for the company is given shown, and a founder would share this address with an employee so that he or she can join the organization. Recall the two choices from Figure 5-4. To join an organization, a user would enter the address given in the Company Address field. Below that, under Ethereum Account, is information on the user connected to this Aragon client. This address matches the one shown at the top right of that page, but here, we also see the remaining balance in that account. You can also rename this account or log out of this company. The more interesting tidbit is shown below the Ethereum Account information, an option to link your account and verify your identity. As we mentioned earlier, not all companies built on Aragon will take the form of a DAO and some might require verification of the participants. Aragon provides access to an Ethereum-Keybase resolver for identity management in specific use cases. Finally, the rest of the features in Aragon Core are accessible through the side menu.

The Settings in Aragon only provide an overview of the user connected to the client, but what's the default state of the company we just created? To get an overview of our DAO, we need to navigate to the Ownership tab shown in Figure 5-6.

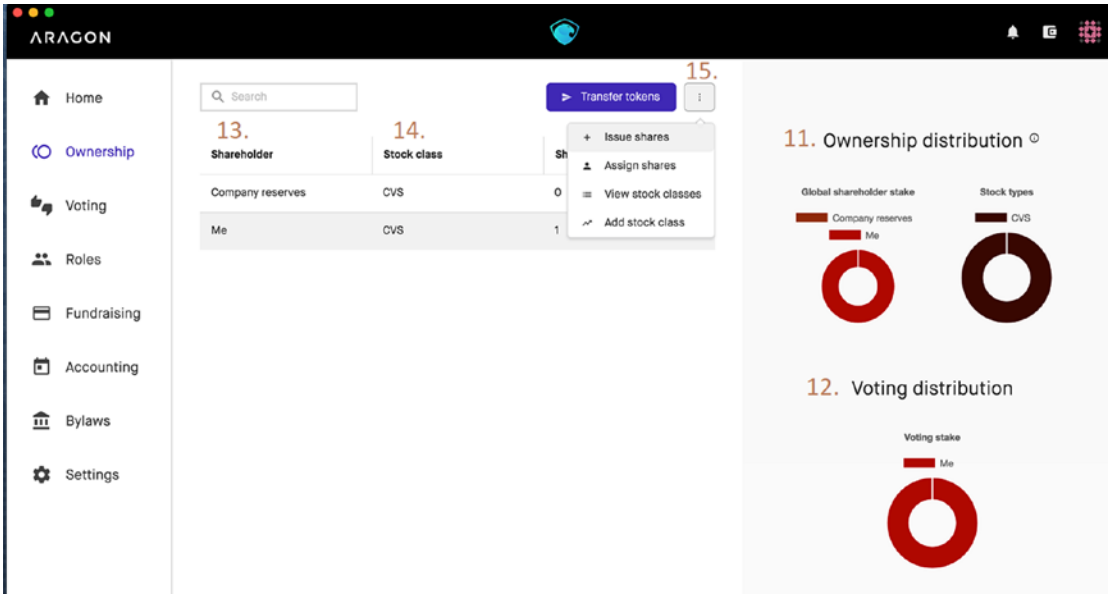


Figure 5-6. *Overview of our DAO*

Essentially, each participant of the DAO will have a stake in decision making, and this page provides the amount of stock each party owns. By default, the creator of a DAO initially owns all the stock. We can see the ownership distribution on the right side of the page, where Me is the current user, and on the pie chart, Me owns 100 percent of the stock. Currently, there is only one type of stock in the template, Class of Voting Stock (CVS), which allows the holder to participate in voting. Generally speaking, the more stock you own, the more influence you can exert on the voting process, but these effects diminish in a DAO with a large number of users. More precisely, the Voting Distribution section shows how the voting power is distributed among the entities in our DAO. We only have one user for now, but this chart will update as new users are added. Next, in this DAO the Shareholder list enumerates the shareholders and currently there are two, the company itself and the creator (executive, "Me"). The only stock class available is the voting stock, to which we will add more soon. Finally, the Ownership tab also allows us to perform advanced functions (e.g., issuing stock) that can be reached through the menu icon in the top right, which is the focus of the second topic in our walkthrough.

Issuing Shares

Now that we have our generic company operational, it's time to start editing the template and customizing the DAO. We explore here how to issue stock, add new types of shares, and assign them to an employee. In Aragon, every action that deals with changing the company's operation relies on members reaching a consensus through majority voting. The default settings give every shareholder voting power, but this can be refined by editing the bylaws. Let's begin by issuing shares to the DAO.

■ **Note** One interesting use case for a company (or a DAO) in Aragon is to be used as a decentralized decision-making body for an open source project or an associated technology. The stocks become a mechanism for decision making for future directions of the project with greater community involvement through voting. The use of tokens and the ability to raise funds also provides a mechanism for providing monetary support to sustain development. We revisit this scenario later in this chapter.

Referring to Figure 5-6, use the menu available in the top right to select Issue Shares to open a screen on the side. Issuing new shares will require us to pick a class of shares to offer and the number of shares to issue. The process is shown in Figure 5-7.

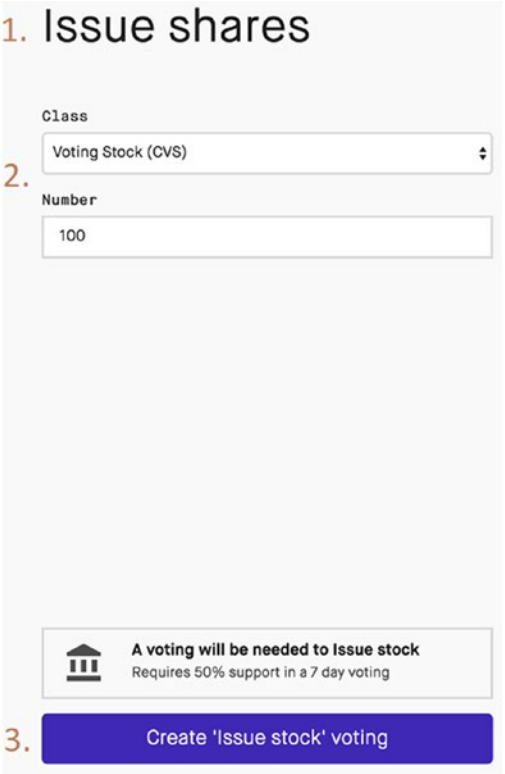


Figure 5-7. *Issuing more shares*

First, we have to decide the class of shares to issue. Currently, there is only one class of shares available, so we will use the default. Next, we choose to issue 100 new shares of the CVS type. To finalize this action, we first have to go through a voting process. Recall that all major changes to the company need to be reviewed by other shareholders and approved before an action can be carried out. Here, additional shares will only be issued if the notion receives at least 50 percent approval from shareholders. Only after that, the company gains more shares that can be reassigned to other users.

Currently there is only one voting user in the DAO, so this will be the deciding vote. The voting interface looks like the general schematic for the voting process shown in Figure 5-8.

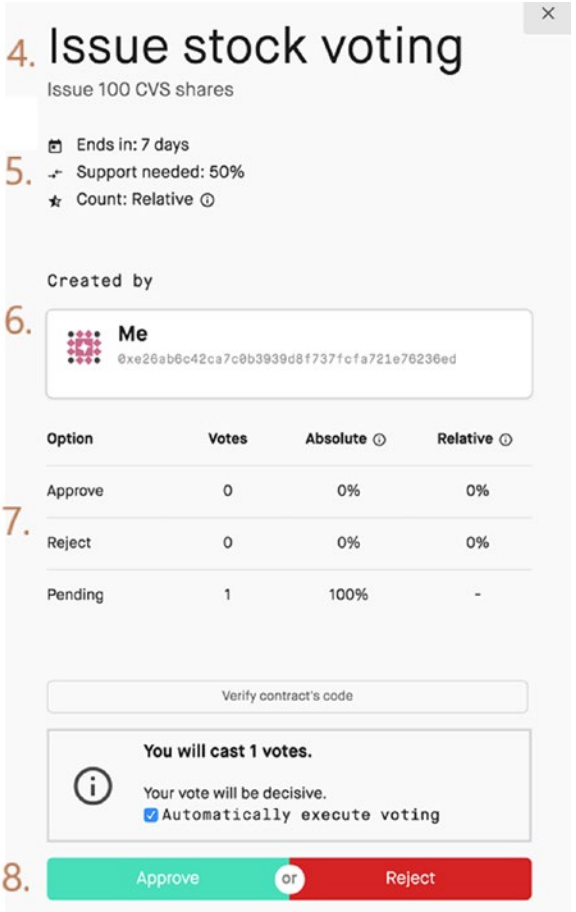


Figure 5-8. *The voting interface*

It follows a very straightforward outline, beginning with the name of the notion bring brought to voters' attention. This is followed by the action being taken in response and a tentative schedule of the voting. The default setting for Aragon is to pass a notion with more than 50 percent approval. The interface next displays who put forth this notion for voting; here Me refers to the user connected to this Aragon client, along with the user's address. The details of voting are shown next. Here only one vote is needed to approve the notion. Finally, the user is presented with two choices: Approve and Reject. After the vote has been cast, it will be gathered in a transaction. When the poll closes, all the votes will automatically be counted and the outcome displayed.

So what does the voting outcome look like? Here, we assume the choice was to approve the notion for our DAO. Figure 5-9 shows the outcome of the vote.

Figure 5-9. *Outcome of voting. The proposition has been accepted and voting is closed. The overall statistics on voting are reported so any user can see how the vote was split*

Issuing more shares just changed the ownership distribution in our DAO. By default, newly issued shares belong to the company and then executives can vote to transfer those shares to other users as they see fit. Figure 5-10 shows the new distribution after shares have been issued. Compare this to Figure 5-6 when the company was just instantiated. Recall that until now, we have only had one stock class, so the global distribution of shares changes but there are no changes in the stock types. Let's add new classes of stocks and explore the options associated with them.

Shareholder	Stock class	Shares
Company reserves	CVS	100
Me	CVS	1

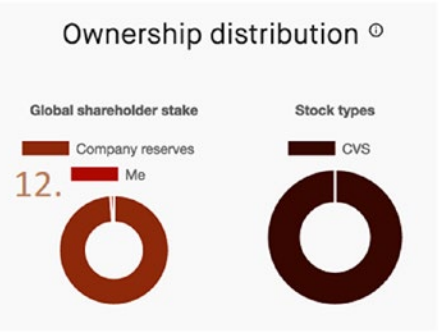


Figure 5-10. *A composite of the changes in ownership distribution with new shares issued*

Notice that the new shares are issued directly to the company, not any particular user. Notice also the changes in the global distribution pie chart. Previously, the Me stake was the only visible portion of the chart. Now, with the new updates, the company holds a majority of the shares.

Before we add new stock types to our DAO, let's briefly talk about the different types of stock typically issued by a public company. There are three major types that we consider here:

- *Common stock (Class A)*: This is what people normally refer to when they mention stocks. Common stock is issued by a company and usually available through stock exchanges such as the New York Stock Exchange (NYSE) or Nasdaq. The main advantages of common stock are payment in dividends and trade in an exchange. A dividend is typically a sum of money paid to the shareholders of a company at the end of a quarter out of the profits generated in that quarter. Common stock can be also be traded or sold at any time on an accredited exchange.
- *Preferred stock (Class B)*: Preferred stock is a special kind of stock issued to investors who have a significant stake in the company. This stock has a few financial advantages over common stock, making it a safer investment. In terms of dividends, preferred stock is cashed before common stock and as a result receives a larger share of a company's profits as compared to common stock. Moreover, common stock can vary in terms of dividends paid out, whereas preferred stock provides fixed returns to the shareholders.
- *Founder stock (Class F)*: Class F shares are an exclusive kind of preferred stock issued to founders of a company. These shares come with greater voting rights; for instance, one share of founder stock might counts for ten votes. In comparison, one share of common stock would only account for one vote. Founder stock grants super voting rights and is normally used to retain control of a company's decision-making process when a large number of entities are involved.

Let's look at the stock classes offered in Aragon. This can be done through the Ownership tab we saw in Figure 5-6. Click Add Stock Class to open the dialog box shown in Figure 5-11.

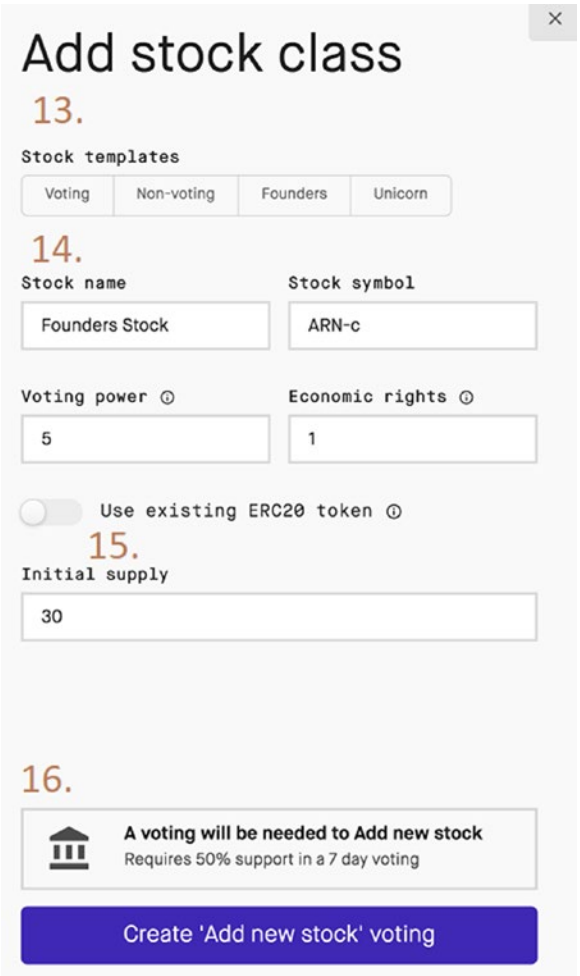


Figure 5-11. *Interface for adding new stock*

There are four stock classes available in Aragon, listed under Stock Templates. Voting stock is the default, non-voting stock may be issued to attract employees to a company where holding stock becomes an economic multiplier, founders stock offers super voting rights and often remains concentrated among the founders, and finally unicorn stock is given to special investors who have a significant stake in the company because it offers a high multiplier and voting power. Next we assign the parameters of our new stock class. We will add a class of founder stock that will be indicated in the ownership with the symbol ARN-c. By default, founder stock has higher voting power than common stock, but that can be updated as necessary. The Economic Rights setting is a multiplier for the stockholder; essentially, the dividends paid back to this stockholder are multiplied by their economic right. Next is the number of shares issued. In this case, we want to add a founder class, and issue 30 new founder shares to the company. As usual, adding a new stock class would require a vote to be cast (16) and then the new shares would be made available.

What does the new stock distribution look like? Figure 5-12 shows the addition of a new stock, and the changes are visible in the Stock Types pie chart.

Shareholder	Stock class	Shares
Company reserves	CVS	100
Me	CVS	1
Company reserves	ARN-c	30

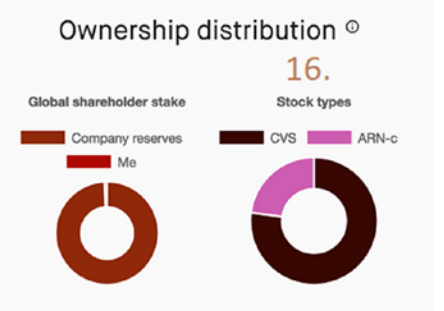


Figure 5-12. *Founder stock added. ARN-c is used as the symbol, and all current shares are owned by the company. To assign shares to members, a vote would have to be taken*

Next, we add another user to our DAO. The second user will be added as an employee and then assigned some shares. We can also show how an existing shareholder can reward another user with exclusive shares for their excellent work or a promotion. To add a new member to your organization, you would have to provide a user with the company address shown in Figure 5-5. They need to download Aragon and join your organization in the first steps after creating a wallet. After a new member joins, he or she can see the current state of the company through the Ownership settings. Figure 5-13 shows the process of assigning a role to a new user after that user has joined your organization.

■ **Note** The terms organization, company, and DAO are being used interchangeably throughout this walkthrough.

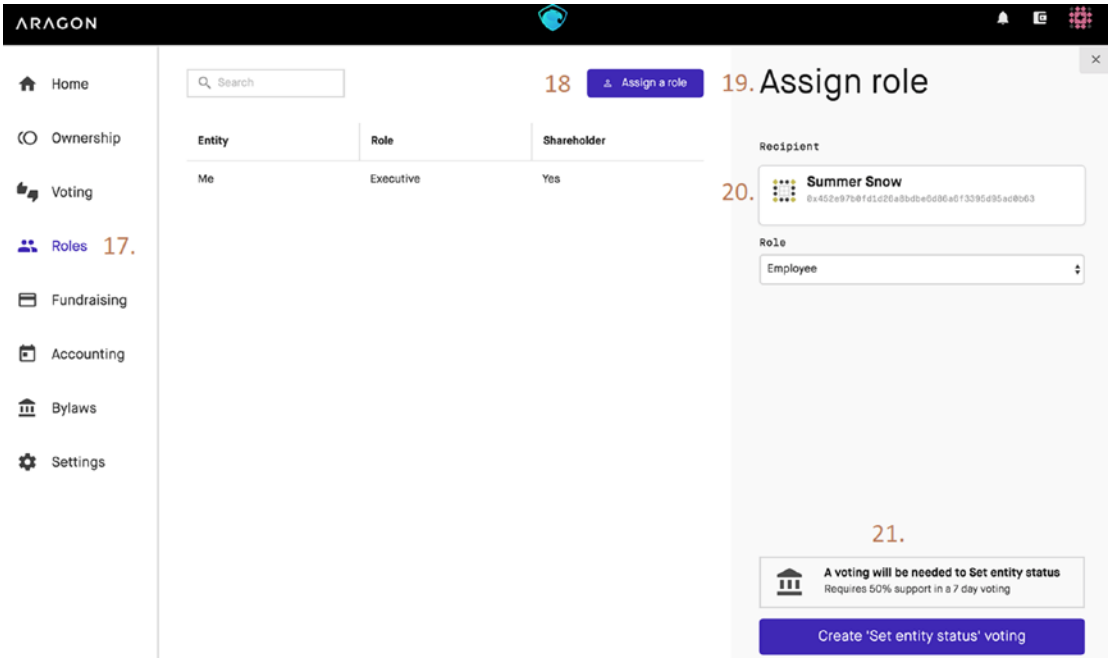


Figure 5-13. *Assigning role to a new employee in the organization*

The Roles tab shown in Figure 5-13 provides an overview of all the entities in the company. Currently, only one user is present: Me is the user connected to this Aragon client. To assign a new role, click Assign a Role to open the Assign Role dialog box. The recipient can be added by their username or an account address in the Recipient field. Depending on your company's policies, if you require some level of identification to be associated with your members, you can add those users by referring to their username. If you don't use the Keybase identity registry, you would have to add new users by referring to an address. After inputting the address, you can decide on the role for a user. Here, we are adding a new employee; however, there are other options available, such as adding a new executive, another employee, or a superuser. The superuser role is normally reserved for nonhuman entities that are a part of your organization, such as oracles or a factory entity that can deploy other components of your company. As always, assigning a new role to a user would require a vote to be cast before the role can be confirmed.

So far, we have discussed how to issue new shares, assign stock, add new stock classes, and assign roles to new members of your company. Here is an exercise for you: Building on the current state of our DAO, add a nonvoting stock class with an economic multiplier of 5, and issue 1,000 shares to the company. Then, assign five of those shares to the new employee, Summer Snow. The result should look like Figure 5-14.

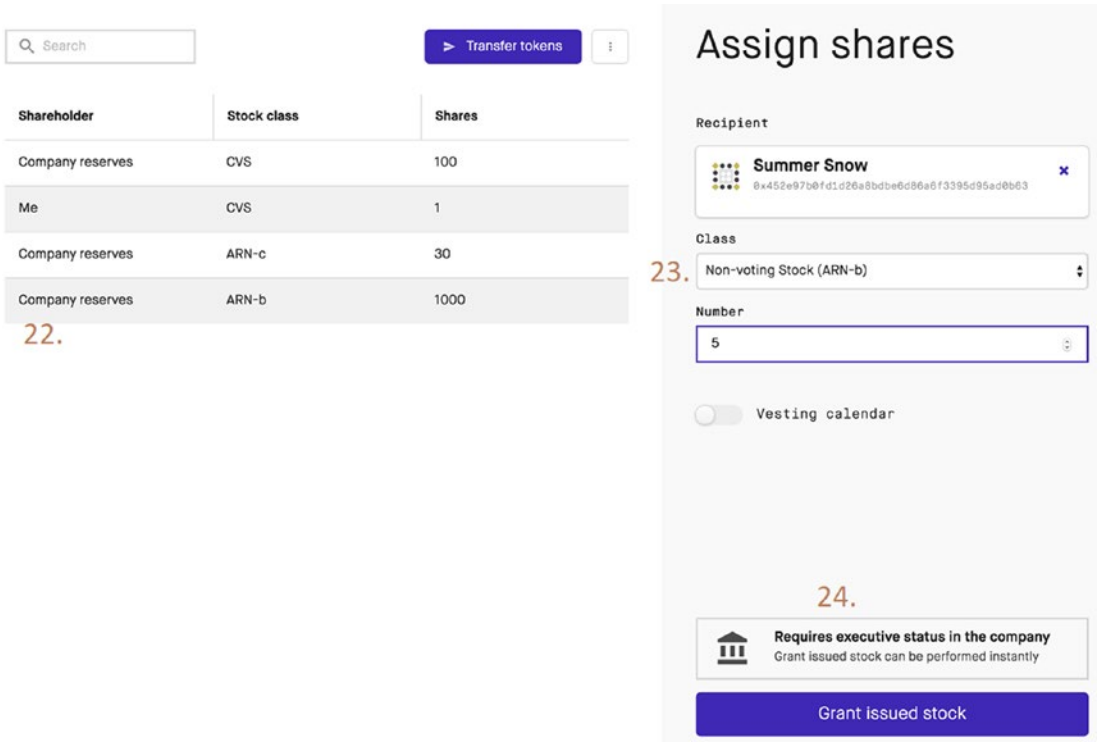


Figure 5-14. *Assigning shares to a new user*

The reader exercise should result in a new stock class called, ARN-b which is a non-voting stock. There should be 1,000 shares issued and by default, all the shares belong to the company. Next, we want to assign shares, which can be done on the Ownership tab. We have to refer to Summer Snow by an address, then pick the stock class and the number of shares to be assigned. To approve this change, we normally need a vote. In this case, however, the non-voting stock has an exception: An executive can grant non-voting stock instantly to another employee without any need for approval. This is limited only to assigning non-voting stock within a company, but this exception can be removed by editing the bylaws.

Here are two more tasks: Assign half of the founders stock to yourself, and transfer 1 founders share to Summer Snow. Both tasks can be performed on the Ownership tab and the final ownership distribution should look like Figure 5-15. The transfer process can also be completely instantly without any voting, but this action is irreversible.

Shareholder	Stock class	Shares
Company reserves	CVS	100
Me	CVS	1
Company reserves	ARN-c	15
Me	ARN-c	15
Summer Snow	ARN-c	1
Company reserves	ARN-b	995
Summer Snow	ARN-b	5

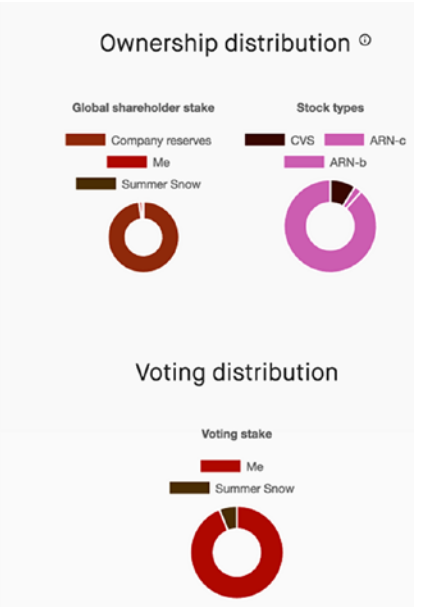


Figure 5-15. *Final ownership distribution at the end of all the steps performed*

The Ownership tab organizes the shareholders by the stock classes they own. To review, we have three classes of stock available now: a non-voting stock (ARN-c), a CVS, and founders stock (ARN-b). The majority of shares with voting power are still owned by the executive, but transferring stock to the employee makes them eligible to participate in voting. On the right side are the newly updated pie charts for voting and global distribution of stock within the company. These common operations allow your company to serve as a governing body and collaboratively determine the future direction of an attached project.

Fundraising and Bylaws

In the last topic of our walkthrough, we handle the more advanced features present in Aragon, namely fundraising on the blockchain and bylaws, which offer another level of fine-tuning for governance of your company. Raising a round of funding on the Aragon blockchain is very different from traditional fundraising. The core idea is simple: Once your DAO is operational, you can issue shares and then you can offer a class of those shares to an investor in return for Ether. These shares can either be offered to an individual investor, or more broadly to the any participant in the network. A round has a specific closing window and starts when it has been approved by the company after voting. The second feature we discuss here is the bylaws, which are the computational equivalent of a configuration file that allows variables to be reassigned and updated as necessary. Bylaws provide a very granular level of access to the governing architecture of your company, and if changes are approved, the new rules are implemented immediately. All the default settings that we have referenced in previous topics arise from a template of bylaws adopted by Aragon to set up a company. Although new bylaws cannot be added to Aragon yet, the existing rules can be changed to create different levels of user permissions for members of the company. Let's look at fundraising in Aragon in Figure 5-16.

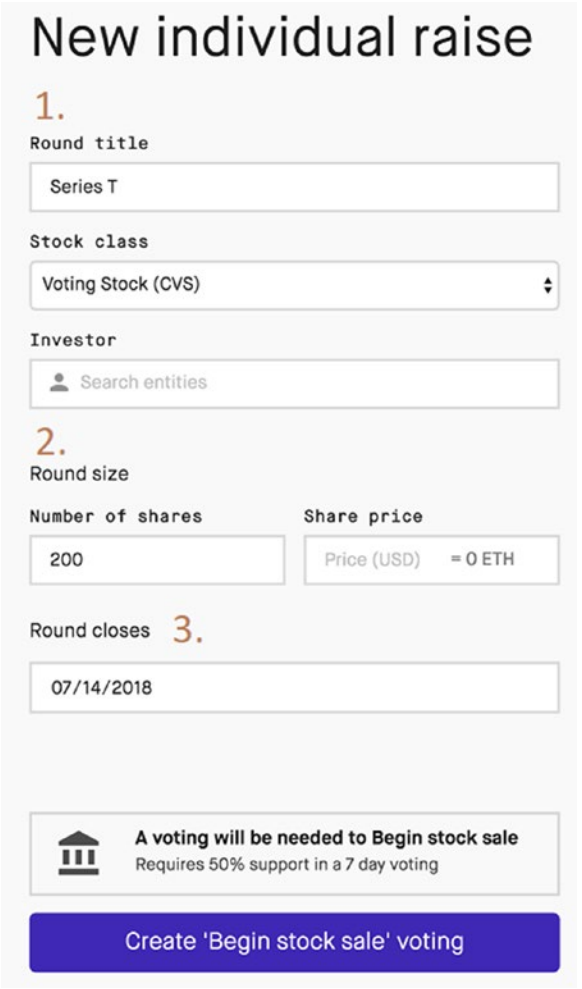


Figure 5-16. *Fundraising from a specific investor in Aragon*

In the Aragon Core, there's a Fundraising tab, where you can click New Raise to open the corresponding dialog box. Currently there are two types of options for fundraising implemented in Aragon. The first one involves raising a round from a specific investor in return for a stake in your company. The second option is a crowdsale, where any participant of the network can buy stock in your company. In this case, we choose the first option and start an individual raise. The series needs a title and the class of shares being offered in return for the funds raised. Normally, an investor would want voting rights, so we picked CVS. After that, you have to enter the investor's address manually and then specify how many shares you want to offer. Once the Aragon network has a true market, the shares can be priced in Ether and have tangible value on the blockchain. The outcome of a project associated with the Aragon-powered DAO will determine the value of the shares in market. The fundraiser has a time-window, at the end of which the round will close. This has the potential to evolve into the all-or-nothing model that Kickstarter uses. Finally, the fundraiser needs to be approved by a majority before it can become active and come online to the network.

Let's look at the bylaws next. Figure 5-17 shows a list of currently implemented bylaws that every entity participating in your company has agreed to honor. Eventually, executive-implemented rules will be allowed in Aragon to offer even more customization over governance.

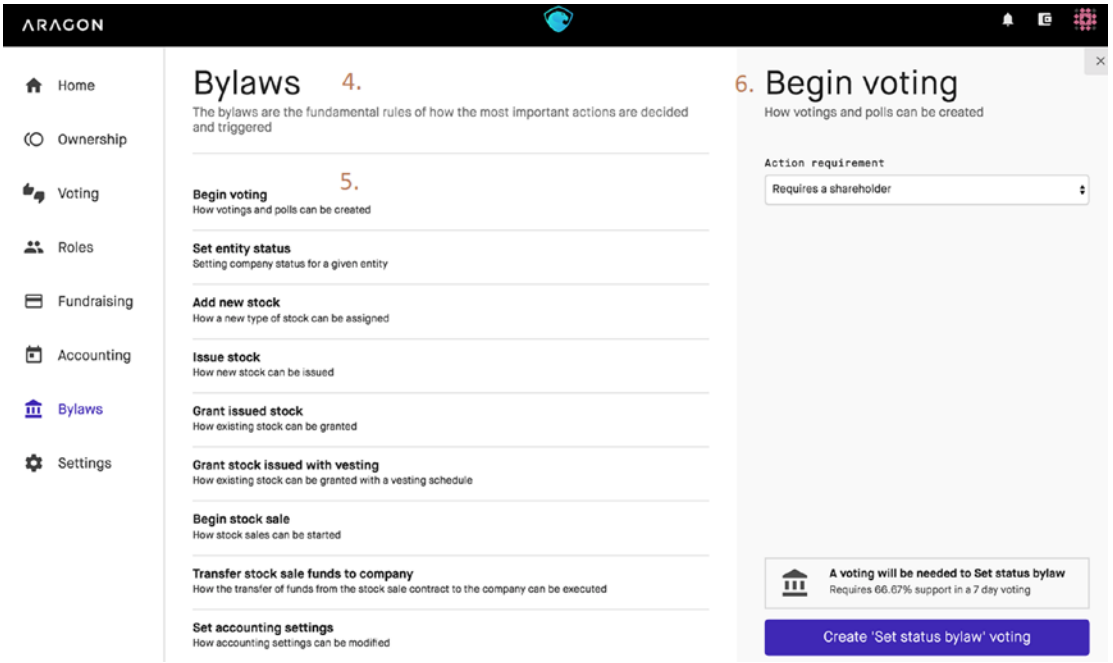

ARAGON
Home
Ownership
Voting
Roles
Fundraising
Accounting
Bylaws
Settings
Bylaws
4.
The bylaws are the fundamental rules of how the most important actions are decided and triggered
5.
Begin voting
How votings and polls can be created
Set entity status
Setting company status for a given entity
Add new stock
How a new type of stock can be assigned
Issue stock
How new stock can be issued
Grant issued stock
How existing stock can be granted
Grant stock issued with vesting
How existing stock can be granted with a vesting schedule
Begin stock sale
How stock sales can be started
Transfer stock sale funds to company
How the transfer of funds from the stock sale contract to the company can be executed
Set accounting settings
How accounting settings can be modified
6. Begin voting
How votings and polls can be created
Action requirement
Requires a shareholder
A voting will be needed to Set status bylaw
Requires 66.67% support in a 7 day voting
Create 'Set status bylaw' voting

Figure 5-17. *Bylaws for the company*

The Bylaws tab lists all the available laws implemented for the company. We take a look here at the first law, which dictates how voting can be initiated. In Aragon, every major action requires a vote from the participants, so changing who can initiate a vote also implies that more actions can be performed that require a vote. The action requirements in the Begin Voting dialog box provide a few options, such as being a shareholder, having a role assigned in the company, or limiting voting to specific addresses (users). By default, voting requires a shareholder, but that can be changed here as needed. As with every other major action, a change in bylaws requires majority approval to be implemented company-wide.

Let's look at another example, this time granting the power to assign roles to new users and how identities can be integrated automatically using an oracle. Figure 5-18 provides a depiction of this example.

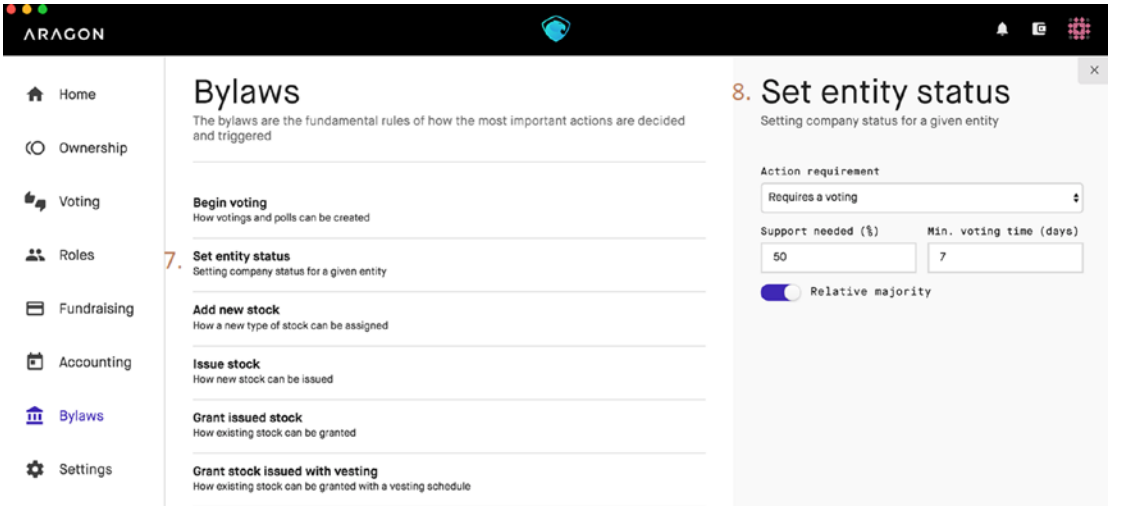


Figure 5-18. *Changing the status of a member in your company*

This process is similar to the previous step, but instead, we will edit the second bylaw, Set entity status, here. In the Set Entity Status dialog box, select the last option: An oracle will be called for confirmation. This changes the parameters, as now you just need to provide the address of the oracle on the network. How does identity management function using an oracle? If your company requires some level of identification, Aragon can be integrated with Keybase and an oracle can confirm the status of identification from Keybase. In this way, new roles can be assigned without the need for a vote and only promotions or entity status updates would require network-wide voting.

Summary

In this chapter, we discussed the concept of decentralized organizations. We started with a presentation of Aragon in the context of decentralized organizations built on the blockchain. We talked about the definition of a DAO and the organs that Aragon implements. We had an in-depth discussion on the identity management features enabled by Aragon and Keybase. Finally, the heart of this chapter was the walkthrough of how to set up, manage, and operate a DAO, including all its basic functions. This chapter has provided the foundation for creating your own DAO.

References

The main reference used to compile this chapter was the Aragon whitepaper and the aragon-core GitHub docs.

CHAPTER 6

The DAO Hacked

Authors: Colin Forward, Vikram Dhillon

It is safe to say that "DAOism" is well on its way to becoming a quasi-cyber-religion.

—Vitalik Buterin[1]

In Chapter 5, we discussed the concept of decentralized organizations and the modus operandi of a DAO. Here, we want to highlight a historic moment leading to the creation of the first DAO, and how it eventually got hacked. Our discussion begins with a fresh perspective on decentralized organizations from Buterin, and leads into the story of Slock.it, the company at the heart of the DAO revolution. Then, we present some code that made The DAO dysfunctional: pieces of the smart contract relevant to the vulnerability, the conditions that allowed repetitive withdrawals from The DAO, and the exploit itself. We conclude the chapter by talking about the consequences of this hack: The debate about hard vs. soft forks, and the creation of Ethereum Classic.

Introduction

Discourse in the global blockchain community has been characterized by idealism going back to Satoshi Nakamoto's early writings on Bitcoin as a response to central banking. The line of reasoning is that systems that are vulnerable to corruption, or in any case cater to the wishes of a select few, could be made more accountable if they were governed by code. If that code lives on the blockchain, then it is impervious to biased intervention by minority parties.

Following that tradition, in a September 2013 blog post for Bitcoin Magazine, Vitalik Buterin explored the notion of the DAO. The article began as follows:

> *Corporations, US presidential candidate Mitt Romney reminds us, are people. Whether or not you agree with the conclusions that his partisans draw from that claim, the statement certainly carries a large amount of truth. What is a corporation, after all, but a certain group of people working together under a set of specific rules? When a corporation owns property, what that really means is that there is a legal contract stating that the property can only be used for certain purposes under the control of those people who are currently its board of directors—a designation itself modifiable by a particular set of shareholders. If a corporation does something, it's because its board of directors has agreed that it should be done. If a corporation hires employees, it means that the employees are agreeing to*

[1]See https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_6

> *provide services to the corporation's customers under a particular set of rules, particularly involving payment. When a corporation has limited liability, it means that specific people have been granted extra privileges to act with reduced fear of legal prosecution by the government—a group of people with more rights than ordinary people acting alone, but ultimately people nonetheless. In any case, it's nothing more than people and contracts all the way down.*
> *However, here a very interesting question arises: do we really need the people?*[2]

Three years after Buterin's article was first published, The DAO came into existence as a smart contract written in Solidity, perhaps the purest manifestation of this idealism. Despite its canonical label, The DAO was not the first—or the last—decentralized autonomous organization. In fact, by May 2016 when the leadership at Slock.it kicked off The DAO's record-breaking initial coin offering (ICO), DAOs were well established as the third wave of the increasingly mainstream blockchain phenomenon.[3]

Although many people consider Bitcoin to be the very first DAO, there were drastic differences in the nature of the two services. Although it is true that Bitcoin was governed by code shared by every miner in the network, Bitcoin doesn't have an internal balance sheet, only functions by which its users can exchange value. Although other DAOs at the time did have a concept of asset ownership, what made The DAO unique was that central to its code were the radically democratic processes that defined how The DAO would deploy its resources. It was a realization of Buterin's concept of a corporation that could conduct business without having a single employee, let alone a CEO.

From the DAO white paper:

> *This paper illustrates a method that for the first time allows the creation of organizations in which (1) participants maintain direct real-time control of contributed funds and (2) governance rules are formalized, automated and enforced using software. Specifically, standard smart contract code has been written that can be used to form a Decentralized Autonomous Organization (DAO) on the Ethereum blockchain.*

Buterin talked about the balance between automation and capital in the context of what sets decentralized organizations apart from traditional companies. Paul Kohlhaas from ConsenSys presented Figure 6-1 to illustrate where DAOs fall on the spectrum of autonomous organizations.

[2]See https://bitcoinmagazine.com/articles/bootstrapping-a-decentralized-autonomous-corporation-part-i-1379644274/

[3]To put this in perspective, 15 days into the DAO's crowdsale, members of the MakerDAO subreddit were discussing proposals that would trigger an investment in MakerDao by the DAO.

DAO Quadrant

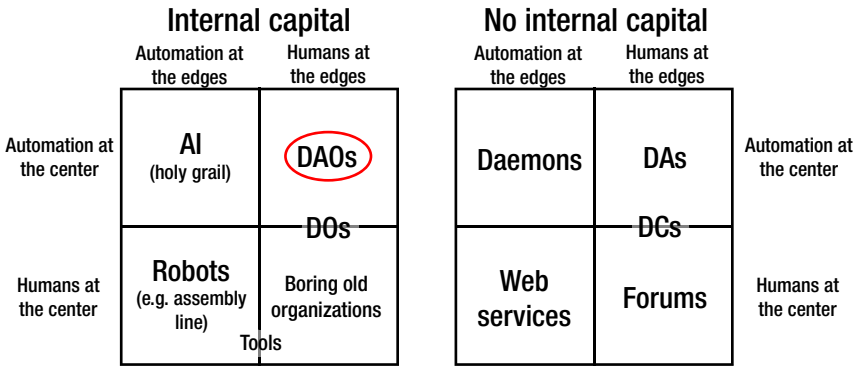


Figure 6-1. *DAOs as automation-powered decision-making entities with human participants*

In essence, DAOs are a paradigm shift from automated entities that previously contained no capital. Using a blockchain allows us to infuse capital and build hybrid business models where we can fine-tune the degree of automation for specific use cases.

The Team

As often is the case in the blockchain world, much confusion surrounds the nature of the relationship between the employeeless DAO and the humans who wrote and maintained The DAO's code. Those humans, in the case of The DAO, were led by the top brass at Slock.it, a German company set on disrupting the sharing economy by way of a technology they called the Universal Sharing Network (USN).

Christopher Jentzsch, Slock.it's CEO, and Stephan Tual, the company's COO, held senior positions at the Ethereum Foundation (Lead Tester and CCO, respectively) prior to starting Slock.it. Jentzsch was the primary developer of The DAO code, and Tual became the face of The DAO via blog posts, conference presentations, and forum contributions. So how would their current company benefit from the creation of a leaderless venture fund built on Ethereum? To understand their motivations we have to examine Slock.it's vision to connect the blockchain to the physical world.

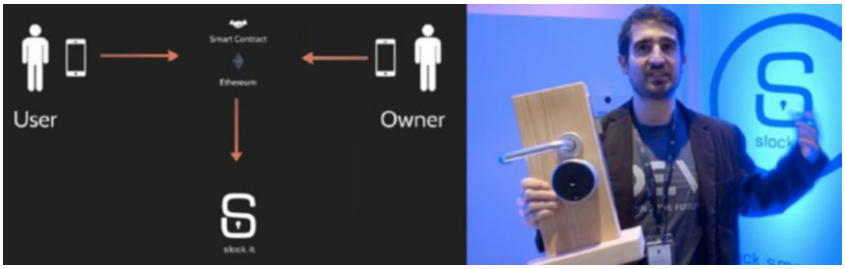
In building the USN, Slock.it set out to play a central role in mainstream adoption of IoT technology. By providing a way to interact with devices on the network from anywhere in the world, the USN would, hopefully, become the backbone of a hyperconnected world, where your property can be rented out to other people without the need for centralized companies like Uber and Airbnb. Instead, the USN would provide an interface to the Ethereum blockchain, where decentralized applications can govern the transactions that make up the sharing economy.

The company intended to build a specialized modem, called the Ethereum Computer, for connecting IoT devices to the USN. Slock.it's vision for The DAO was to create a decentralized venture fund that would invest in promising proposals to build blockchain-supported products and services.

At the time of writing (18 months after The DAO's ICO), Slock.it has raised $2 million in seed funding to continue developing the USN and the Ethereum Computer. According to Tual's blog posts on the company web site, Slock.it will now make the Ethereum Computer available as a free and open source image for popular system-on-a-chip (SoC) systems such as Raspberry Pi. The company also built and supports Share&Charge, a service that lets owners of electric vehicle charging stations sell their power to electric vehicle owners via a blockchain-powered mobile app.

Jeremiah Owyang from Crowd Companies summarized one of the main use cases of Slock.it in a slide shown in Figure 6-2.

Slock.it smart locks link to secure Ethereum contracts on the blockchain



When someone purchases a Slock, it's connected to the Slock smart contract in the Ethereum blockchain and controlled by it. The owner of a Slock can set a deposit amount and a price for renting his property, and the user will pay that deposit through a transaction to the Ethereum blockchain, thereby getting permission to open and close that smart lock through their smart phone.

Figure 6-2. *Slock.it can act as a decentalized Airbnb by linking the purchase of a physical device (a smart lock) to a smart contract*

Ultimately, this idea was expanded to become a decentralized IoT platform, where any device could be connected to the blockchain.

The DAO

The original conception of The DAO was not the radical experiment in democratic business processes that was eventually released at its ICO. Jentzsch described the process on the Slock.it blog:

> *In the beginning, we created a slock.it specific smart contract and gave token holders voting power about what we—slock.it—should do with the funds received.*
>
> *After further consideration, we gave token holders even more power, by giving them full control over the funds, which would be released only after a successful vote on detailed proposals backed by smart contracts. This was already a few steps beyond the Kickstarter model, but we would have been the only recipient of funds in this narrow slock.it-specific DAO.*

> *We wanted to go even further and create a "true" DAO, one that would be the only and direct recipient of the funds, and would represent the creation of an organization similar to a company, with potentially thousands of Founders.*[4]

To achieve this decoupling of Slock.it and The DAO, Jentzsch designed a Solidity contract that would allow any DAO token holder to make proposals for how The DAO's resources should be handled. All token holders could vote on active proposals, which had a minimum voting period of 14 days.

That meant that once The DAO's ICO was complete, Slock.it would have to submit a proposal to The DAO just like anyone else. Other users could use the Mist browser to evaluate the proposal. This is how proposals were structured:

```
struct Proposal {
    address recipient;
    uint amount;
    string description;
    uint votingDeadline;
    bool open;
    bool proposalPassed;
    bytes32 proposalHash;
    uint proposalDeposit;
    bool newCurator;
    SplitData[] splitData;
    uint yea;
    uint nay;
    mapping (address => bool) votedYes;
    mapping (address => bool) votedNo;
    address creator;
}
```

As you can see, proposals—the core of this automaton code base that would quickly raise $150 million—were relatively simple requests for The DAO's resources (`uint` amount).

Any DAO token holder could vote on proposals by calling the `vote` function:

```
function vote(
    uint _proposalID,
    bool _supportsProposal
) onlyTokenholders returns (uint _voteID);
```

The votes from any one address would be weighted proportional to the amount of DAO tokens held at that address. If token holders wanted to vote for two separate positions, they could transfer the amount of tokens they wanted to vote with to another address, and vote again from there.[5] Any tokens voting on an open proposal were locked (could not be transferred) until the end of the voting period.

The uint proposalDeposit was the deposit (in wei) that creators of a proposal had to stake on the proposal until the voting period closed. If the proposal never reached quorum, the deposit would remain with The DAO.

---

[4] https://blog.slock.it/the-history-of-the-dao-and-lessons-learned-d06740f8cfa5

[5] This could be the case because proposals required a quorum of 20 percent of votes to have weighed in on a proposal for the vote to be valid.

There were two special types of proposals that did not require deposits that would play a key role in the fate of The DAO. The first type was a proposal to split The DAO, effectively withdrawing the funds of the recipient of the proposal into a new "child" DAO, which was a clone of the original, but at a new contract address. Split proposals had a voting period of 7 days, instead of 14, and anyone who voted yes on a split proposal would follow the recipient, withdrawing their tokens from the original DAO and moving them into the resultant child DAO.

The second special type of proposal was to replace the curator of The DAO. DAO curators were addresses set at the creation of The DAO and the creation of any child DAO that could whitelist recipient addresses, effectively serving as gatekeepers.[6] If the majority vote no on a proposal to replace a curator, the yes votes can elect to stand by their decision, creating a new DAO with their chosen curator.

The ICO Highlights

The ICO for the initial DAO concept was an overnight success:

- It raised 12 million ETH (~$150 million).
- Both Jentzsch and Tual admitted that they never expected their idea to be so successful.

The Hack

The idea that The DAO was vulnerable to attack had been floating around in the developer community. Vlad Zamfir and Emin Gün Sirer first raised the issue in a blog post calling for a moratorium on The DAO until the vulnerabilities could be addressed.[7] Just days before the attack, MakerDAO cautioned the community that their code was vulnerable to an attack, and Peter Vessenes demonstrated that this vulnerability was shared by The DAO.[8]

These warnings prompted a now infamous blog post published on June 12, 2016 by Tual on the Slock.it web site titled "No DAO funds at risk following the Ethereum smart contract 'recursive call' bug discovery." Within a couple of days, fixes had been proposed to correct many of the known vulnerabilities of The DAO, but it was already too late. On June 17, an attacker began draining funds from The DAO.

The DAO attacker exploited a well-intentioned although poorly implemented feature of The DAO that was intended to prevent tyranny of the majority over dissenting DAO token holders. From The DAO white paper:

> *A problem every DAO has to mitigate is the ability for the majority to rob the minority by changing governance and ownership rules after DAO formation. For example, an attacker with 51% of the tokens, acquired either during the fueling period or created afterwards, could make a proposal to send all the funds to themselves. Since they would hold the majority of the tokens, they would always be able to pass their proposals.*
>
> *To prevent this, the minority must always have the ability to retrieve their portion of the funds. Our solution is to allow a DAO to split into two. If an individual, or a group of token holders, disagrees with a proposal and wants to retrieve their portion of the Ether before the proposal is executed, they can submit and approve a special type of proposal to form a new DAO. The token holders who voted for this proposal can then split the DAO, moving their portion of the Ether to this new DAO, leaving the rest alone only able to spend their own Ether.*

[6]Curators weren't necessarily human gatekeepers. Gavin Wood "resigned" as a curator of The DAO to make a point that curation was merely a technical role and that the curator had no proactive control over The DAO.
[7]http://hackingdistributed.com/2016/05/27/dao-call-for-moratorium/
[8]http://vessenes.com/more-ethereum-attacks-race-to-empty-is-the-real-deal/

Unfortunately, the way that this "split" feature was implemented made the DAO vulnerable due a catastrophic reentrancy bug.[9] In other words, someone could recursively split from the DAO, withdrawing amounts equal to their original ETH investment indefinitely, before the record of their withdrawal was ever recorded in the original DAO contract.

Here is the vulnerability, as found in the Solidity contract file `DAO.sol`:

```
function splitDAO(
  uint _proposalID,
  address _newCurator
) noEther onlyTokenholders returns (bool _success) {

  ...
  // [Added for explanation] The first step moves Ether and assigns new tokens
  uint fundsToBeMoved =
      (balances[msg.sender] * p.splitData[0].splitBalance) /
      p.splitData[0].totalSupply;
  if (p.splitData[0].newDAO.createTokenProxy.value(fundsToBeMoved)(msg.sender) == false) //
[Added for explanation] This is the line that splits the DAO before updating the funds in
the account calling for the split

  ...
  // Burn DAO Tokens
  Transfer(msg.sender, 0, balances[msg.sender]);
  withdrawRewardFor(msg.sender); // be nice, and get his rewards
  // [Added for explanation] The previous line is key in that it is called before
totalSupply and balances[msg.sender] are updated to reflect the new balances after the split
has been performed

  totalSupply -= balances[msg.sender]; // [Added for explanation] This happens after the
split
  balances[msg.sender] = 0; // [Added for explanation] This also happens after the split
  paidOut[msg.sender] = 0;
  return true;
}
```

As shown here, The DAO referenced the `balances` array to determine how many DAO tokens were available to be moved. The value of `p.splitData[0]` is a property of the proposal being submitted to the DAO, not any property of the DAO. That, in combination with the fact that `withdrawRewardFor` is called before `balances[]` is updated, made it possible for the attacker to call `fundsToBeMoved` indefinitely, because their balance will still return its original value.

[9]Rentrancy is a characteristic of software in which a routine can be interrupted in the middle of its execution, and then be intiated (reentered) from its beginning, while the remaining portion of the original instance of the routine remains queued for execution.

Looking more closely at `withdrawRewardFor()` shows us the conditions that made this possible:

```
function withdrawRewardFor(address _account) noEther internal returns (bool _success) {
  if ((balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply < paidOut[_
account])
    throw;

  uint reward =
    (balanceOf(_account) * rewardAccount.accumulatedInput()) / totalSupply - paidOut[_
account];
  if (!rewardAccount.payOut(_account, reward)) // [Added for explanation] this is the
statement that is vulnerable to the recursion attack. We must go deeper.

    throw;
  paidOut[_account] += reward;
  return true;
}
```

Assuming the first statement evaluates as false, the statement marked as vulnerable will run. There's one more step to examine to understand how the attacker was able to make that the case. The first time the `withdrawRewardFor` is called (when the attacker had legitimate funds to withdraw), the first statement would correctly evaluate as false, causing the following code to run:

```
function payOut(address _recipient, uint _amount) returns (bool) {
  if (msg.sender != owner || msg.value > 0 || (payOwnerOnly && _recipient != owner))
      throw;
  if (_recipient.call.value(_amount)()) { // [Added for explanation] this is the coup de
grace
      PayOut(_recipient, _amount);
      return true;
  } else {
      return false;
}
```

`PayOut()` as written in the second `if` statement references "`_recipient`"—the person proposing the split. That address contains a function that calls `splitDAO` again from within `withdrawRewardFor()`, before the token balance at that address is updated. That created a call stack that looked like this:

```
splitDao
      withdrawRewardFor
         payOut
            recipient.call.value()()
               splitDao
                 withdrawRewardFor
                    payOut
                       recipient.call.value()()
```

The attacker was therefore able to withdraw funds from The DAO into a child DAO indefinitely.

To recap, the attacker accomplished the following:

1. Split the DAO.
2. Withdraw their funds into the new DAO.
3. Recursively called the split DAO function before the code checked to determine if the funds were available.

This process is visually represented in Figure 6-3.

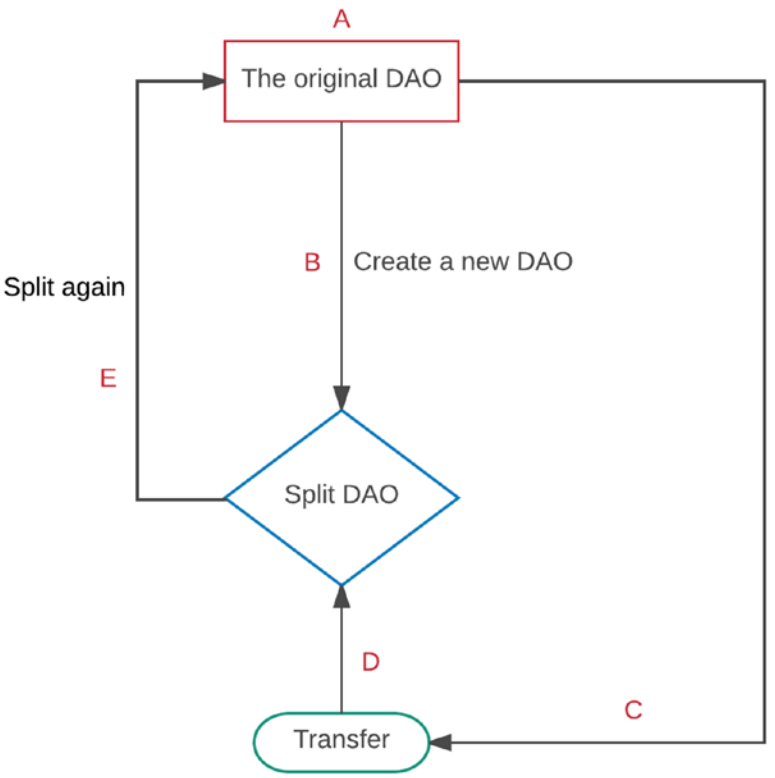


Figure 6-3. *The process of iterative withdrawal*

In Figure 6-3, we can see the iterative process visually. The original DAO is represented by A, and a sub-DAO is created in B. Then, a transfer function requests some funds be withdrawn from the original DAO in C. Finally, the funds are transferred to the new DAO created. This process is repeated again as new DAOs are created with each loop.

The Debate

The result was that the attacker was able to steal about 3.6 million ETH, worth about $50 million at the time of the attack. The DAO investors were left in an especially precarious position. Not only had The DAO been compromised, but if they tried to withdraw the funds into their own child DAO, the resulting contract would have the same vulnerabilities as the original.

However, The DAO investors weren't the only ones with an interest in the outcome of this turn of events. The hype surrounding the DAO had reached the religious proportions predicted by Buterin back in 2014. Almost 5 percent of ETH in circulation at the time was invested in The DAO. That had a number of implications for the entire Ethereum ecosystem, and led to one of the most contentious debates in the short history of blockchains.

On one side of the debate were those looking to protect the fledgling Ethereum ecosystem from a malicious actor in possession of a nontrivial portion of the total ETH in circulation. They were not necessarily concerned with whether or not The DAO would survive, but ultimately wanted to ensure that Ethereum would survive as a reputable blockchain platform on which other DAOs could be built in the future. This was the disposition of Buterin and many of the core members of the Ethereum development team.

On the other side were those committed to the ideals of decentralization and immutability. In the eyes of many in this camp (we'll call them the justice camp), the blockchain is an inherently just system in that it is deterministic, and anyone choosing to use it is implicitly agreeing to that fact. In this sense, the DAO attacker had not broken any laws. To the contrary, the reentrancy attack used the software code that made up The DAO's bylaws and turned it against itself.

The decentralization camp believed that rewriting the blockchain to roll back the attacker's sequestration of ETH in child DAOs would compromise the integrity of the blockchain. The blockchain, according to this line of thought, was supposed to be immutable and without any central authority, including the Ethereum Foundation. They were concerned the moral hazard of a small group of people rewriting the blockchain could open the door to other interventions, such as selective censorship.

The two sides debated their positions passionately over social media and in news outlets. The process made famous the concepts of soft forks and hard forks. Forking blockchains—or any software code for that matter—was not new to Ethereum or The DAO, but it became the focus of the debate between the justice camp and the immutability camp.

Meanwhile, a group of white hat hackers were working around the clock to try and hack the hacker. The white hat group consisted of people both for and against the hard fork, but they worked together, nonetheless, to perform some of the same attacks that had been identified before June 17 to move the stolen ETH into new contracts in hopes of returning it to its rightful owners.[10]

The white hat team reached out to people who had made significant investments in The DAO to raise money for stalking attacks, in which they could follow the attacker into new DAOs with greater funds than the attacker was able to withdraw, giving them majority voting rights in the resulting DAO.

The Split: ETH and ETC

On July 30, over 90 percent of the hashrate signaled its support for the fork. The DAO funds were returned to investors as though the organization had never existed. Sort of.

Opposition to the hard fork led to the emergence of Ethereum Classic (ETC), as a small portion of the community continued to mine the original Ethereum blockchain. These immutability fundamentalists were committed to the idea that blockchains represented a new, disruptive governance model.

The most visible member of the movement was Arvicco, a Russian developer using a pseudonym. In a July 2016 interview with Bitcoin Magazine, he characterized the disagreement in this way:

> *By bailing out the DAO, the Ethereum Foundation is attempting to reach a shortsighted goal of "making investors whole" and "boosting confidence in Ethereum platform." But they're doing quite the opposite. Bailing out the DAO undermines two of the three key long-term value propositions of the Ethereum platform.*[11]

[10] https://www.reddit.com/r/ethereum/comments/4p7mhc/update_on_the_white_hat_attack/
[11] https://bitcoinmagazine.com/articles/rejecting-today-s-hard-fork-the-ethereum-classic-project-continues-on-the-original-chain-here-s-why-1469038808/

Despite the tenacity of this vocal minority in the Ethereum community, many people did not expect both versions of the blockchain to survive long term. Major exchanges and cryptoservices added support for ETC but many were skeptical of the long-term prospects of a platform that essentially duplicated Ethereum's capabilities.

Erik Vorhees, founder and CEO of Shapeshift.io, expressed skepticism about ETC's ability to remain relevant, but explained that, ultimately, he believed that the split was good for the blockchain ecosystem. In November 2016 he told Decentralize Today:

> *While this caused quite a bit of turmoil (still ongoing), it's hard for me to say it was a failure. A division within the community has now been resolved, and since both camps were significant enough in size, we now have two Ethereums, for a while at least. It has actually made the community more peaceful, because instead of the two camps arguing over who is right, both of them can be "right" in their own way, and the market will decide whose product is actually better. I expect ETH to win out over ETC, but I have to admit ETC has survived longer than I thought.*

At the time of this writing, ETC continues to grow as a platform and as a community. Despite ETC appreciating less rapidly than ETH, BTCC and Huobi recently announced that they would be adding the token to their exchanges. ETC developers have also accelerated their departure from Ethereum as a platform, with the release of Mantis, the first client built from scratch for ETC (as opposed to Ethereum's Mist, Parity, and other clients).

The Future

When a technology fails after being hyped in a massive spotlight, it is incredibly difficult to recover the credibility of the ideas powering that technology. What does the future of DAOs look like? Any user investing in a DAO token should be cautious, but there have been massive security advances to a DAOs' structure and governance. Interestingly, Paul also presented a new outlook of DAOs as the next-generation of automated VCs called Decentralized Fund Managers. According to Paul, DAOs represent a new class of financial asset management tools where a software can manage a fund that would normally be entrusted to traditional venture capitalists. By implementing software based management at its core, any profits made by a DAO are distributed directly to the token holders. The members of this new DAO are essentially investors, and they would be issued a new class of tokens that represent their holdings (or stake) and earnings. Ultimately, in a DAO, the members can guide how the funds are being allocated and what benefits are offered in return for investment. It would stand to reason that a DAO managing funds would operate in traditional VC cycles:

- The transition first cycle involves investing using the ETH funds
- The second cycle involves the management of a DAO into a next-generation automated VC. The governance model can provide for new decision-making abilities for early investors such as angel-syndicates.

We discuss the idea of artificial intelligence (AI) leading financial investments in the final chapter of the book.

Summary

The future of Ethereum is bright despite the DAO hack. With the emergence of Ethereum Classic and the incredible rate of new developments, the platform is pushing closer to maturity. It must be noted that Ethereum as a platform was not the cause of the vulnerability. In its nascent state, smart-contract code is bound to cause bugs such as this hack, which will result in better code-checking mechanisms and secure code-writing practices that can avoid such pitfalls. In the future, as a result of forks, we might end up with a consolidated single-currency platform just like before.

CHAPTER 7

Ethereum Tokens: High-Performance Computing

In the Ethereum ecosystem, transfer of value between users is often realized by the use of tokens that represent digital assets. Ether is the default token and the de facto currency used for transactions and initializing smart contracts on the network. Ethereum also supports the creation of new kinds of tokens that can represent any commonly traded commodities as digital assets. All tokens are implemented using the standard protocol, so the tokens are compatible with any Ethereum wallet on the network. The tokens are distributed to users interested in the given specific use case through an ICO. In this chapter, we focus our attention on tokens created for a very specific use case: high-performance computing (HPC). More precisely, we discuss a model of distributed HPC where miners offer computational resources for a task and get rewarded in some form of Ethereum tokens.

We begin our discussion with an overview and life cycle of a token in the network. Then, we dive into the first token, Ethereum Computational Market (ECM), which is the most generalized and comprehensive distributed processing system. ECM will be our standard model for HPC using tokens, and we will introduce concepts such as dispute resolution (firm and soft) and verifiability of off-chain processing with on-chain rewards. The second token we cover is Golem, which posits itself as the Airbnb for unused CPU cycles. The first release, called Brass Golem, will allow distributed rendering of 3D objects using Blender. Future releases will enable more advanced functionality such as processing big-data analytics. The third token we present is Supercomputing Organized by Network Mining (SONM), which extends the idea of fog computing to create an Ethereum-powered machine economy and marketplace. SONM has published a more technically oriented roadmap focused on neural networks and artificial intelligence. The initial use case would be to provide decentralized hosting to run a Quake server. The last token we talk about in this chapter is iEx.ec, which leverages well-developed desktop grid computing software along with a proof-of-computation protocol that will allow for off-chain consensus and monetization of resources offered.

Tokens and Value Creation

To understand the concept of tokens, we first need to understand the context in which they operate, "fat" protocols. At Consensus 2017, Naval Ravikant talked about the idea of "fat" protocols and the fundamental differences between Internet companies and the next generation of startups building on the blockchain. Currently, the Internet stack is composed of two slices, a "thin" slice of protocols that power the World Wide Web and a "fat" slice of applications that are built on top of the protocols. Some of the largest Internet companies such as Google and Facebook captured value in the application layer, but then had to invent new protocols and an infrastructure layer to actually scale. When Internet companies reached that size, they had validated their core business model and had enough resources to allocate toward creation of new protocols.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017

V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_7

Blockchain-based companies operate on a different stack with a "thin" slice of application layer and a "fat" slice of protocol layer. The value concentrates in the protocol layer, and only a fraction spills over to the application layer. Joel Monégro and Balaji S. Srinivasan proposed two probable reasons for the large interest and investment in the protocol layer:

- *Shared data layer*: In a blockchain stack, the nature of underlying architecture is such that key data are publicly accessible through a blockchain explorer. Furthermore, every member of the network has a complete copy of the blockchain to reach consensus on the network. A pertinent example of this shared data layer in practical usage is the ease with which a user can switch between exchanges such as Poloniex and Kraken. or vice-versa. The exchanges all have equal and free access to the underlying data or the blockchain transactions.
- *Access tokens*: Tokens can be thought of as analogous to paid API keys that provide access to a service. In the blockchain stack, the protocol token is used to access the service provided by the network, such as file storage in the case of Storj. Historically, the only method of monetizing a protocol was to build software that implemented the new protocol and was superior to the competition. This was possible for research divisions in well-funded companies, but in academia, the pace of research was slower in the early days because the researchers creating those protocols had little opportunity for financial gain. With tokens, the creators of a protocol can monetize it directly through an ICO and benefit more as the token is widely adopted and others build services on top of the new protocol.

Due to proper incentives, a readily available data sharing layer, and application of tokens beyond the utility of a currency, developers are spending considerable time on the underlying protocols to crack difficult technical problems. As a result, startups building on the blockchain stack will inevitably spend more time on the "fat" protocol layer and solve technical challenges to capture value and differentiate themselves from a sea of Ethereum tokens.

Recently, with more tokens up and coming in the Ethereum ecosystem, cross-compatibility has become a concern. To address this issue, a new specification called ERC20 has been developed by Fabian Vogelsteller. ERC20 is a standard interface for tokens in the Ethereum network. It describes six standard functions that every access token should implement to be compatible with DApps across the network. ERC20 allows for seamless interaction with other smart contracts and decentralized applications on the Ethereum blockchain. Tokens that only implement a few of the standard functions are considered partially ERC20-compliant. Even partially compliant tokens can easily interface with third parties, depending on which functions are missing.

■ **Note** Setting up a new token on the Ethereum blockchain requires a smart contract. The initial parameters and functions for the token are supplied to the smart contract that governs the execution of a token on the blockchain.

To be fully ERC20-compliant, a developer needs to incorporate a specific set of functions into their smart contract that will allow the following actions to be performed on the token:

- *Get the total token supply:* `totalSupply()`
- *Get an account balance:* `balanceOf()`
- *Transfer the token:* `transfer(), transferFrom()`
- *Approve spending of the token:* `approve(), allowance()`

When a new token is created, it is often premined and then sold in a crowdsale also known as an ICO. Here a *premine* refers to allocating a portion of the tokens for the token creators and any parties that will offer services for the network (e.g., running full nodes). Tokens have a fixed sale price. They can be issued and sold publicly during an ICO at the inception of a new protocol to fund its development, similar to the way startups have used Kickstarter to fund product development.

The next question we should ask regarding tokens is this: Given that tokens are digital, what do token buyers actually buy? Essentially, what a user buys is a private key. This is the analogy we made to paid API keys: Your private key is just a string of a characters that grants you access. A private key can be understood to be similar to a password. Just as your password grants you access to the e-mail stored on a centralized database like Gmail, a private key grants you access to the digital tokens stored on a decentralized blockchain stack like Ethereum.

■ **Tip** The key difference between a private key and a password stored on a centralized database is that if you lose your private key, you will not be able to recover it. Recently, there have been some attempts to restore access to an account through a recovery service using a cosigner who can verify the identity of the user requesting recovery.

Ultimately, tokens are a better funding and monetization model for technology, not just startups. Currently, even though base cryptocurrencies such as Ethereum have a larger market share, tokens will eventually amount to 100 times the market share. Figure 7-1 provides a summary of the differences between value creation for traditional Internet companies and companies building on the blockchain stack using Ethereum tokens. Let's begin with our first token, which will serve as a model for all the tokens that follow. Michael Oved at Consensys suggested that tokens will be the killer app that everyone has been waiting for:

> *If killer apps prove the core value of larger technologies, Ethereum tokens surely prove the core value of blockchain, made evident by the runaway success that tokens have brought to these new business models.*
>
> *We are now seeing a wave of launches that break tokens through as the first killer app of blockchain technology. Bitcoin, in many ways, is a "proof of concept" for a blockchain-based asset. The Ethereum platform is proving that concept at scale.*

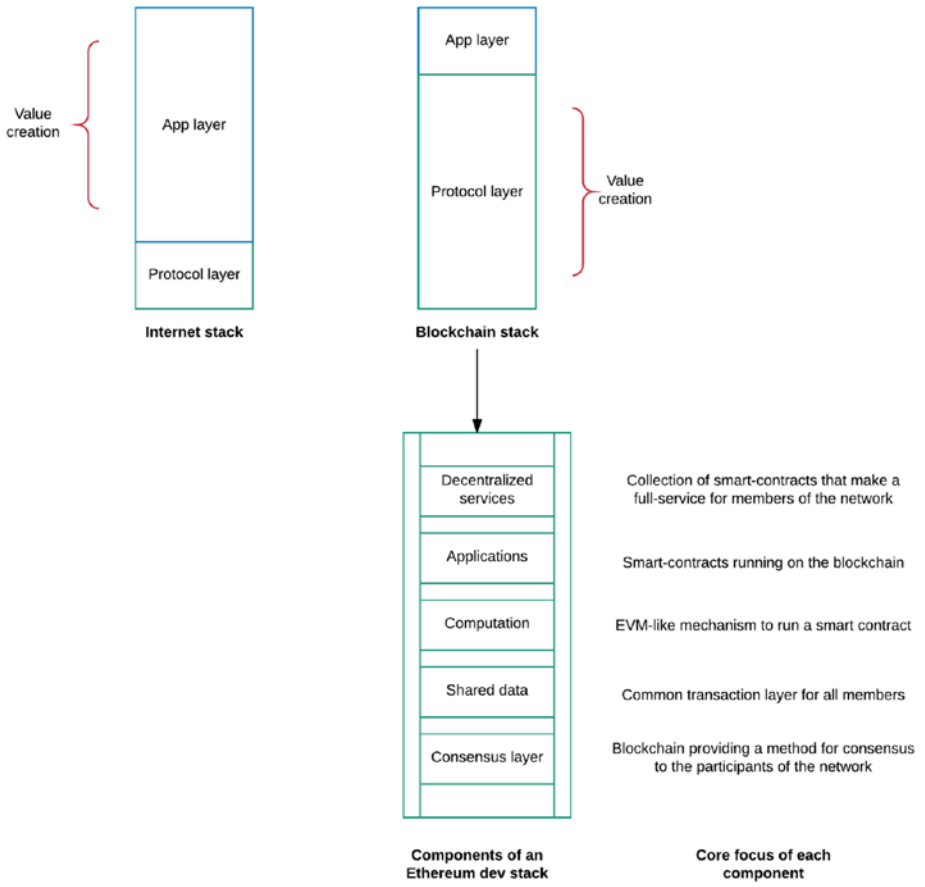


Figure 7-1. *An overviwew of a blockchain stack used by tokens*

This model is based on Joyce J. Shen's description of distributed ledger technology. The traditional Internet companies such as Google and Facebook created and captured value in the application layer by harvesting data that users generated. Blockchain companies have a different dynamic in terms of ownership and the technology available. The blockchain itself provides a mechanism for consensus and a shared data layer that is publicly accessible. Ethereum further provides a Turing-complete programming language and node-to-node compatibility through an EVM that can run the instructions contained in a smart contract. Using smart contracts, a full application can be constructed to run on the blockchain and a collection of applications becomes a full decentralized service such as Storj.

Ethereum Computational Market

ECM is a very simplistic and functional token for off-chain computations and the first token we consider in this chapter. It serves as a general model covering all the necessary features that an HPC token would need. ECM is designed to facilitate execution of computations off-chain that would be too costly to perform within the EVM. In essence, ECM is a decentralized version of Amazon's EC2. The key technical advance in ECM is the computation marketplace, which allows one user (the customer) to pay another (the host) for executing an algorithm off-chain and report the results back to the customer. Additionally, to preserve the integrity of the computation being done off-chain, each algorithm also has an on-chain implementation. This on-chain component can be used to verify whether the submitted end result is indeed correct. This on-chain implementation also plays a role in resolving disputes. Let's take a look at the life cycle of a request through ECM in Figure 7-2. ECM uses some terminology specific to the project for describing the life cycle of a request when it's initially received to the final resolution. We employ some of this terminology in Figure 7-2, along with a few additional terms:

- *Pending*: Indicates when a request is received. Every request begins in the pending status.
- *Waiting for resolution*: A request for computation is submitted to the network, and a host computed the algorithm and is now reporting the result. This is a decision point for the customer: Either the answer will be accepted and the request moves to soft resolution status, or the answer is challenged, in which case the request moves to needing resolution state.
- *Needs resolution*: When an answer is challenged, this path is taken from the decision tree. The request is changed to needs resolution status and on-chain verification component of the algorithm is required.
- *Resolving*: This is the interim period as on-chain computation is being performed for a request. The request will remain in this status until the computation has been completed.
- *Firm vs. soft resolution*: Once the on-chain computation has been completed, the request is set to the firm resolution status. If no challenges are made within a certain window of time after the answer has been submitted to the customer, the request is set to soft resolution.
- *Finalized*: Once an answer is obtained either through soft or hard resolution, the original request can be set to finalized status. This unlocks the payment from the customer's end and allows the host to receive payment for off-chain computation.

■ **Note** The reader should understand that Ethereum Computational Markets is presented here as a generic model to highlight all the components of a HPC token. That's why we go through the technical details of computing requests, on and off-chain processing, market dynamics and other concepts. In the later sections, all the generic components are replaced by well-thought-out mechanisms and features.

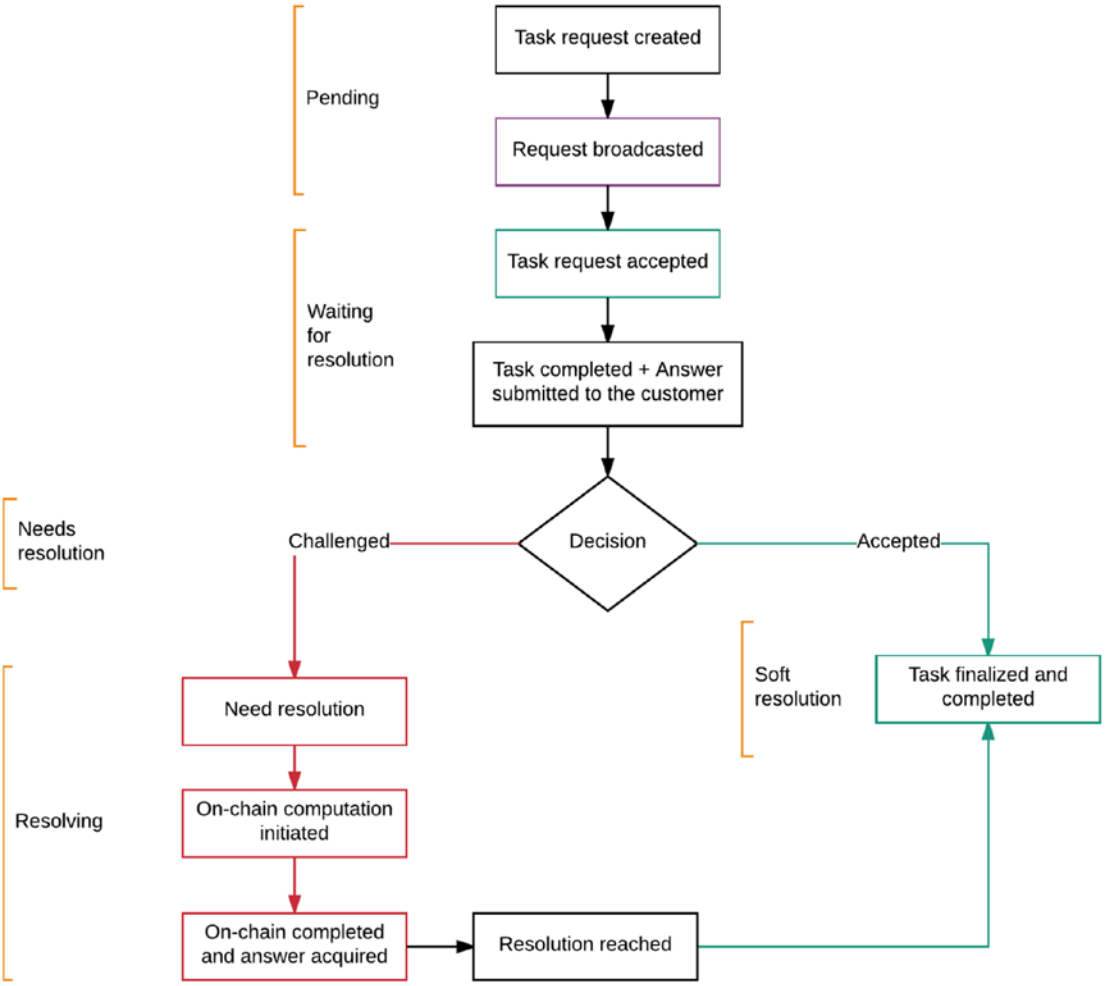


Figure 7-2. *Life cycle of a request as it goes through ECM*

Each request state just described is broken down further into the workflow here. Once a task request is processed off-chain, there is a decision point concerning the final answer. If the customer challenges the answer provided by the host, the on-chain component of the algorithm will execute and the request will go through a few states of needing resolution. On the other hand, if no challenges are made, the request moves to soft resolution and is considered finalized. The host receives payment after the request has reached the finalized state.

Now that we have an understanding of how a request gets processed in ECM, let's talk about the marketplace and individual markets within that marketplace. The concept that we have referred to as an *algorithm* being computed off-chain is made from three separate contracts: a broker contract, a factory contract, and an execution contract.

- *Broker*: A contract that facilitates the transfer of a request from the customer to the host who will carry out the computation.
- *Execution*: A contract used for on-chain verification of a computation. This contract can carry out one cycle of on-chain execution in the event that the submitted answer is challenged.
- *Factory*: A contract that handles the deployment of the execution contract on-chain in the case of dispute resolution. This contract supplies relevant metadata such as the compiler version required to recompile the bytecode and verify it.

The marketplace itself has verticals (markets within a marketplace) that execute specialized types of contracts and algorithms, creating use-case-specific HPC economics on the Ethereum blockchain using tokens. Figure 7-3 provides a visual summary of the three contracts that are a component of each market. What is required to make a computation request on ECM? There are primarily two functions that create a request and provide all the necessary details. Here, the `requestExecution` function is used to create a request for computation and this function takes two arguments. First is a set of inputs to the function itself and second is the time window during which a submitted answer can be challenged before the request turns to soft resolved status. This time window is given in number of blocks, because blocks are created at a definitive time interval in Ethereum. Finally, this function also specifies the payment being offered for this computation. The `getRequest` function returns all the metadata regarding the request. The following are some of the relevant parameters returned from the metadata:

- `address requester`: The address of the customer that requested the computation.
- `bytes32 resultHash`: The SHA3-hash of the result from a given computation.
- `address executable`: The address of the executable contract that was deployed on-chain to settle a challenged answer.
- `uint status`: The status of a request, at a given time through the life cycle. It is an unsigned integer corresponding to the status of a request given by numbers 0 through 7.
- `uint payment`: The amount in wei that this request will pay (to the host) in exchange for completion of a computation. Wei is the smallest unit of Ether that can be transferred between users, similar to Satoshis in Bitcoin.
- `uint softResolutionBlocks`: A time window given by number of blocks within which a challenge to an answer must be submitted. If no challenges are submitted to an answer in that interval, the request changes to soft resolution.
- `uint requiredDeposit`: Amount in wei that must be provided when submitting an answer to a request, or by a challenger objecting a submitted answer. This deposit is locked until the request moves to finalized state and then is released back to the host.

It is important to note that in our discussion, a few tasks such as answer verification, conflict resolution, and challenges have required deposits. The deposits are used as a countermeasure to reduce the potential for Sybil attacks on the network. A Sybil attack is an attack where a single adversary is controlling multiple nodes on a network, and now, the adversary can manipulate PoW on the network.

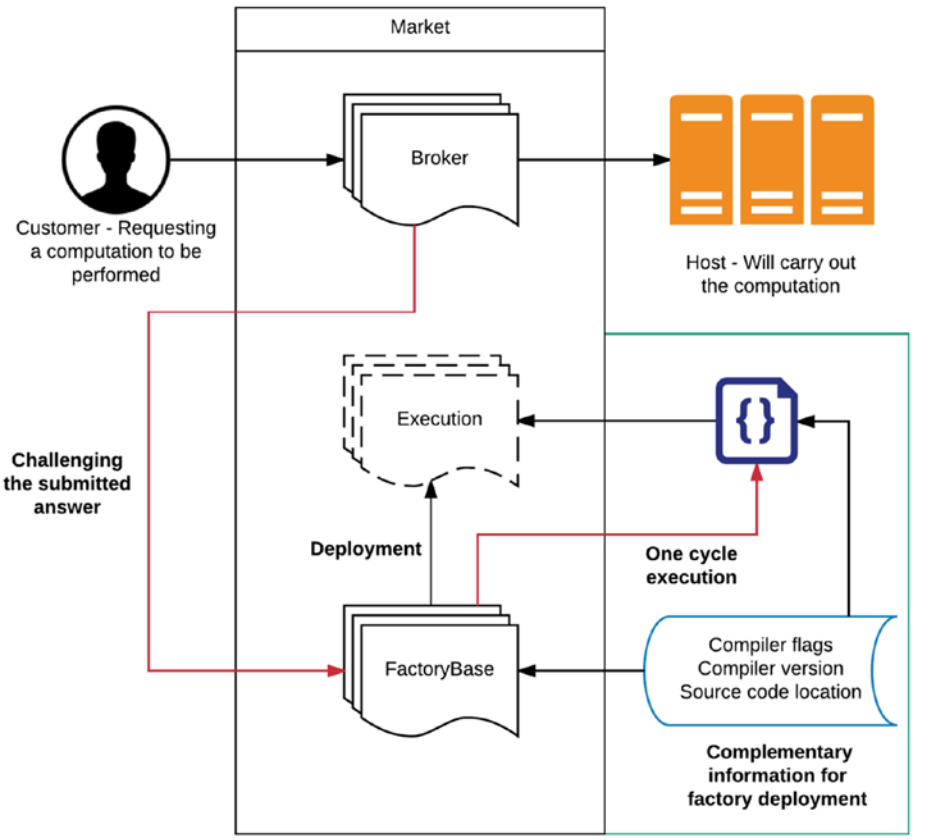


Figure 7-3. *Overview of a market in the marketplace*

The three components in a market necessary for responding to a request by a user (customer) are the broker contract, an execution contract, and a factory contract. The broker contract facilitates a customer–host interaction and the other two contracts play a role in dispute resolution. In the case of challenges to a submitted answer, the broker will initiate the deployment of the execution contract through its factory contract (also called FactorBase, shown with the red workflow), which consolidates the information and prepares for one-cycle execution of the algorithm. The gas necessary for execution will be taken from a deposit that the challenger is required to submit. We discuss the submission and challenge process next.

Next, we talk about the submitting an answer back to the customer and the challenge process. Submitting an answer for a computation done off-chain is performed with the `answerRequest` function. This function takes a unique ID of the request being answered as an input. The actual submission of a potential answer requires a deposit in Ether. This deposit is locked until the request has reached either a soft or hard resolution state. We discuss the importance of holding deposits from involved parties shortly. Once a request has been finalized, the deposit made by the host who submitted an answer is reclaimed, along with the reward for performing the computation. If a submitted answer does not fall within the expectations

of the request submitted by a customer, a participant can challenge the answer. This initiates an on-chain verification process that will execute one cycle of the computation within the EVM to verify whether the submitted answer is correct. If the submitted answer was found to be incorrect during on-chain computation, the host's deposit will have had the gas costs of that computation deducted from it. The challenger will get a large portion of the reward from the customer's deposit, and the dispute will be resolved.

For a request where a submitted answer has been challenged, the request moves to the needs resolution state. This is accomplished by calling the `initializeDispute` function, which serves as a transition between the point at which a challenge is made and the beginning of on-chain verification. Now the broker contract will use a factory to deploy an executable contract initialized with the inputs for this request. The gas costs for calling this function and performing one-cycle execution are reimbursed from the challenger's deposit. Throughout the resolving state on a request, the `executeExecutable` function is called until the on-chain verification has been completed. At this point, the request is moved to a hard resolution state and eventually finalized. The gas charges and reward system might seem complicated, but it follows a simple principle: The correct answer receives payment for the computation, and incorrect submitters must pay for gas during on-chain verification. Let's recap:

- In the case of soft resolution, a host reclaims the initial deposit and a reward given by the customer for executing the computation.
- If there were no correct submitted answers, the gas costs for verification are split evenly among the users who submitted answers. The reward payment returns to the customer who originally requested the computation.
- In the case of hard resolution, the incorrect host reclaims the remaining deposit after gas costs have been deducted (for on-chain verification). This host does not receive a reward for the computation.
- If a challenger wins hard resolution, they reclaim their deposit along with the reward payment. The gas costs are debited from the incorrect host's deposit.

On-chain verification of a computation is by nature an expensive task due to gas costs. ECM is designed such that off-chain computation would be cheaper than running a task in EVM. From a technical standpoint, there are two types of on-chain execution implementations: stateless and stateful computations. Figure 7-4 shows a contrast between the two implementations.

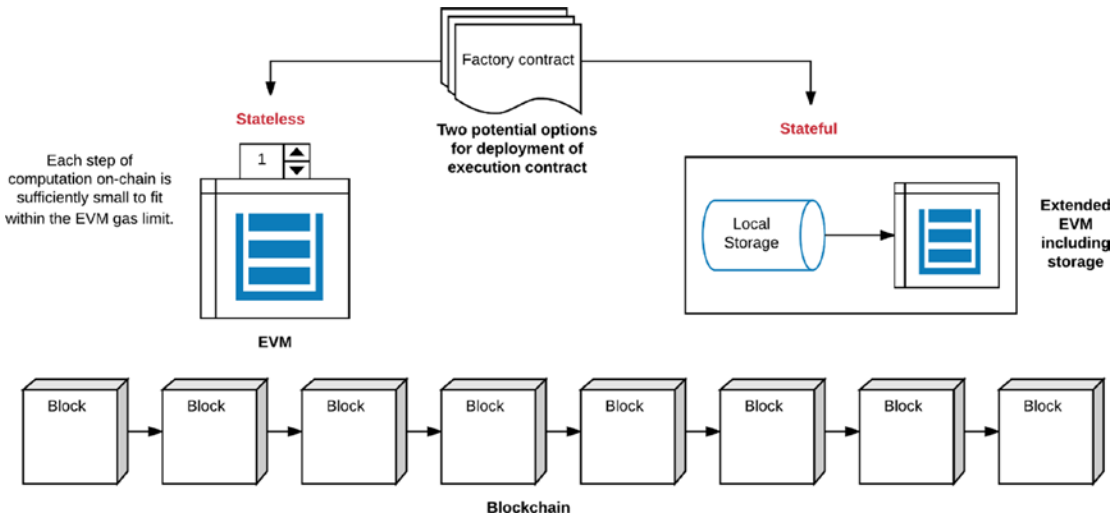


Figure 7-4. *Two models for factory execution contracts in ECM*

An executable contract is stateless if the computation is self-sufficient in that it does not require external sources of data. Essentially, the return value from a previous step is used as the input for the next step. Piper Merriam (the creator of ECM) proposed that stateless executable contracts are superior to stateful implementations for two main reasons: lower overhead while writing the algorithm and reduced complexity because the computation cycles are self-sustaining. An example highlighted by Piper is the Fibonacci sequence written as a stateless implementation with the algorithm returning the latest Fibonacci numbers as the input for the next cycle of execution. The overhead of writing this in a stateless form is very minor and there is no additional complexity of introducing local storage.

■ **Tip** For stateless contracts, the execution contract is identical to algorithm sent as a request to the marketplace for computation. Here, on-chain verification would run the execution contract for one cycle and hosts would run it for as many cycles as necessary to obtain the final answer.

An executable contract is stateful if the computation is not self-sufficient in that it requires an additional data source and the return values from the previous step. This additional data source is often in the form of a data structure holding local storage. Let's go back to our example of the Fibonacci sequence and make it stateful. To do this, the algorithm would store each computed number in the contract storage and only return the latest number to the algorithm for the next cycle of execution. Every step of execution would require the algorithm to search the last number stored to compute the next number. By including storage, now the algorithm will search through saved results and print out any desired Fibonacci sequence that has been computed. This reliance on local state is what makes this instance of the contract stateful. Stateful contracts also enable new and complex features such as using lookup tables and search, but also cause an increase in the complexity of the written algorithm.

During on-chain verification, a single cycle of the execution contract will be executed. However, a self-contained contract in stateless form will be executed efficiently and without any additional complexity. In stateless contracts, each step executed inside an EVM is elementary so it falls within the gas limits of that virtual machine. On the other hand, some contracts require additional storage due to the complexity of execution. In such cases, a stateful contract is executed where local storage is bound to the EVM during the on-chain verification. This storage is temporary and only exists through the duration of the on-chain processing.

Golem Network

Golem (https://golem.network/) is a decentralized general-purpose supercomputing network. In addition to being a marketplace for renting computational resources, Golem aims to power microservices and allow asynchronous task execution. As a technology stack, Golem offers both Infrastructure-as-a-Service (IaaS) and Platform-as-a-Service (PaaS) through a decentralized blockchain stack for developers. In Golem, there are three components that function as the backbone of a decentralized market:

- *Decentralized farm*: A mechanism to send, organize, and execute computation tasks from individual users known as requesters and hosts known as providers (of computational resources). This provides the users competitive prices for tasks such as computer-generated imagery (CGI) rendering and machine learning on network access.
- *Transaction framework*: Golem has a custom payment framework for developers to monetize services and software developed to run on the decentralized network. This framework can be customized to capture value in new and innovative methods such as escrows, insurance, and audit proofs on the Ethereum blockchain.
- *Application registry*: Developers can create applications (for particular tasks) that take advantage of Golem as a distribution channel and a marketplace with new and unique monetization schemes. These applications can be published to the application registry, which essentially functions as an app store for the network.

Golem's premise in building the marketplace is that not all members will request additional resources for extensive computation at all times. As such, the requesters can become providers and rent their own hardware to earn extra fees. Figure 7-5 provides an overview of how the three components in Golem work in sync. Currently, Golem inherits the integrity (Byzantine fault tolerance) and consensus mechanisms from Ethereum for the deployment, execution, and validation of tasks. However, eventually Golem will use fully functional micropayment channels for running microservices. This will allow users to run services like a note-taking app, website hosting, and even large-scale streaming applications in a completely decentralized manner. Several more optimizations are needed before Golem reaches a level of maturity, and currently the most pressing developments are concerning execution of tasks. Before Golem can execute general-purpose tasks, we need to ensure that the computation takes place in an isolated environment without privileges, similar to an EVM but expanded with more features. We also need whitelist and blacklist mechanisms along with digital signatures recognized in the application registry that allow providers to build a trust network and for users to run applications cryptographically signed by trusted developers. Additionally, a network-wide reputation system is necessary to reward the providers that have participated the most during computation tasks and also to detect a malicious node and mitigate tasks efficiently.

■ **Note** The first release dubbed Brass Golem will only allow one type of task to be executed on the network, CGI rendering. The main goals of this release are to validate the task registry, and basic task definition scheme. The developers want to integrate IPFS for decentralized storage, a basic reputation system for the providers involved, and docker integration in the network.

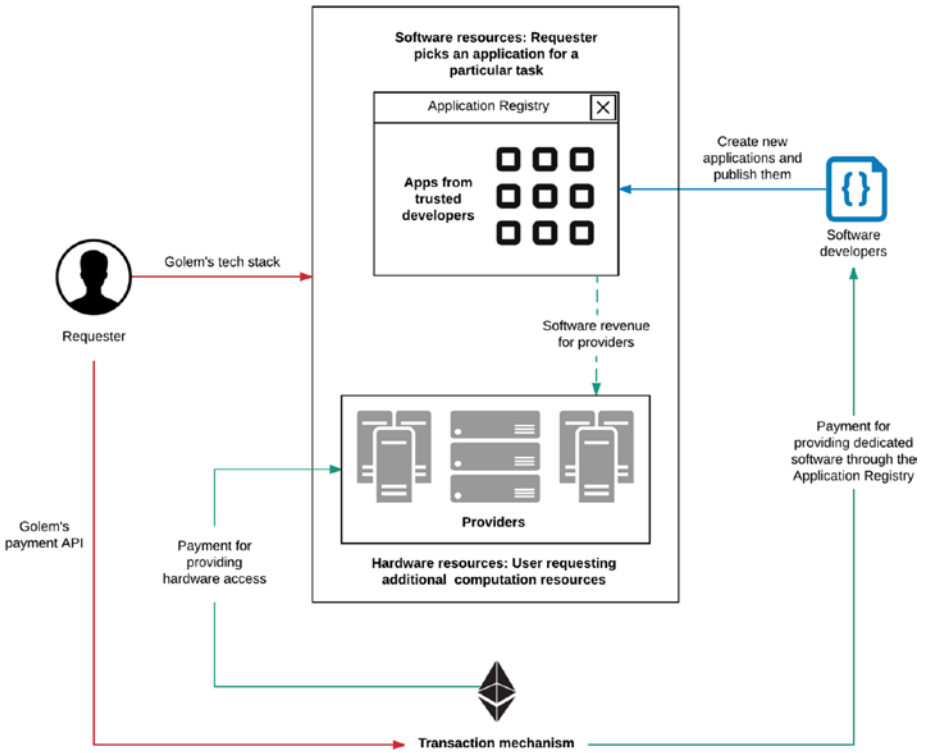

Software resources: Requester picks an application for a particular task
Application Registry
Apps from trusted developers
Create new applications and publish them
Software developers
Golem's tech stack
Requester
Software revenue for providers
Payment for providing dedicated software through the Application Registry
Providers
Golem's payment API
Payment for providing hardware access
Hardware resources: User requesting additional computation resources
Transaction mechanism

Figure 7-5. *The Golem technology stack*

Three main components of the Golem stack are shown in Figure 7-5. Software developers create applications that automate routine tasks and publish them on the application registry. For instance, a time-limited rendering application can eliminate the need for any complicated transactions or task tracking. The buyer will simply make a single-use deposit and use the application until the time is up. When buyers use applications from the registry, the revenue generated is used to compensate the developers for their app. The registry contains two types of applications: verified apps from trusted developers and new apps from unconfirmed developers. Eventually verifiers pool in the newly uploaded applications and track them closely for any suspicious activity. After a few uses, the new applications also achieve verified status. The developers also receive minor compensation from the execution of these applications on the provider nodes. Providers requisition the hardware for apps and the lion's share of revenue generated from running a task is given back to the providers (miners). The transaction mechanism assures the proper delivery of funds to the appropriate party and the buyer is charged for task execution by the same mechanism.

Application Registry

The application registry provides requesters with a massive repository to search for specific tools or applications fitting their needs. The registry is an Ethereum smart contract among three entities: the authors, validators, and providers. What's the design rationale behind adding these three entities? For general-purpose computing, the code is isolated in a sandbox and then executed with the bare minimum privileges. Potential software bugs could wreak havoc in a provider's sandbox, however, executing malicious code in a virtual machine with the intention of escalation of privileges, or even take over the machine. That's why

sandboxing alone is not enough for Golem. One could ask whether the code can be evaluated automatically for safety and security. Theoretically, this is not plausible. We cannot determine the outcome of a complex algorithm before executing it due to the halting problem.

■ **Note** The halting problem is the problem of determining whether a program will finish running or continue to run forever given an arbitrary computer program and an input.

To enable secure and trusted code execution on host machines, the application registry is split among three parties that are responsible for maintaining integrity of the network. Authors publish decentralized applications to the registry, validators review and certify DApps as safe by creating a whitelist, and providers maintain blacklists of problematic DApps. Validators also maintain blacklists by marking applications as malicious or spam, and providers often end up using a blacklist from validators. Similarly, providers can preapprove certain Golem-specific applications (certified by validators) to execute on their nodes.

Providers can also elect to curate their own whitelists or blacklists. The default option for Golem is to run using a whitelist of trusted applications. The first set of apps will be verified by the developers to kickstart Golem, however, after the initial distribution, the network will rely on validators. On the other hand, providers can also take the approach of relying only on blacklists. This approach has the advantage of maximizing the reach of a provider (to the marketplace) and offering a wide range of applications that can be executed on a node. Eventually, providers will become specialized and fine-tune their nodes to be incredibly efficient at running one kind of task. This allows the providers more control over what runs on their nodes and custom hardware options for a computing farm. Ultimately, this option is available to developers who want to maximize their profits and are willing to run dedicated machines with bleeding-edge software.

In a decentralized network like Golem, we will see a divide between traditional and vanguard validators. Traditional validators will maintain a set of stable applications that perform very routine tasks. For instance, a request for a complex 3D rendering can launch a preemptible rendering instance on a provider's nodes. Once the job is completed, the instance will terminate to free up memory and send the output to the requester. On the other hand, vanguard validators will include and approve software that pushes the boundaries of what is possible with Golem. Running new and experimental software on a provider's node will require better sandboxing and hardening of the virtual machine. But these features can be a premium add-on running on special nodes offered by a provider. Monetizing the premium add-ons will also disincentivize scammers and malicious entities. Overall, this design approach makes the network more inclusive, costly for scammers, and innovative for developers.

Transaction Framework

The transaction framework can be considered analogous to Stripe, an API for monetizing applications running on Golem. After the crowdsale, the network will use Golem Network Token (GNT) for all transactions between users, to compensate software developers and computation resource providers. The transaction framework is built on top of Ethereum, so it inherits the underlying payment architecture and extends it to implement advanced payment schemes such as nanopayment mechanisms and transaction batching. Both innovations mentioned here are unique to Golem, so let's talk about why they are necessary to the network. To power microservices, Golem will have to process a very high volume of small transactions. The value of a single payment is very low and these payments are also known as nanopayments. However, there is one caveat when using nanopayments: The transaction fees cannot be larger than the nanopayment itself. To solve this problem, Golem uses transaction batching. This solution aggregates nanopayments and sends them at once as a single Ethereum transaction to reduce the applied transaction fees. For instance, Golem developers note that the cost of ten payments processed in a single transaction is approximately half the cost of ten payments processed in ten transactions. By batching multiple transactions, a significantly lower transaction fee will be passed on to a user paying for per-unit usage (per-node or per-hour) of a microservice.

■ **Note** For providers, another model to power microservices is credit-based payment for per-unit hosting. Here, the requester makes a deposit of timelocked GNT and at the end of the day, the provider automatically deducts the charges for usage. The remaining credits are released back to the requester.

In Golem, nanopayments work in the context of one-to-many micropayments from a requester to many providers that assist in completing the computational tasks. The payments carried out for microservices are on the scale of $0.01 and for such small sums, the transaction fees are relatively large even on Ethereum. The idea here is that instead of making individual transactions of $0.01, the requester (payer) issues a lottery ticket for a lottery for a $1 prize with 1/100 chance of winning. The value of such a ticket is $0.01 for the requester and the advantage is that on average only one ticket in 100 will lead to an actual transaction. This is a probabilistic mechanism to allow nanotransactions, but it does not guarantee that one given Golem node will be compensated adequately if the number of tasks computed is small. Bylica et al. provided the mathematical background to assuring fair distribution of income from this lottery system as the network expands, adding more requesters and providers. Essentially, as the number of tasks by requesters increases, the income a node generates from probabilistic lottery rewards will approach the amount it would receive if being paid with individual transactions.

The lottery system to be implemented in Golem is much more predictable for the providers than a completely probabilistic scheme. The provider is assured that among the tickets issued to reward the providers of a single task, there are no hash collisions and only one ticket is winning. Moreover, if there are 100 providers participating in the lottery payment, then the nanopayment protocol guarantees that the task would only have to be paid out once. Bylica and collaborators discussed a few counterclaim mechanisms in place to prevent malicious entities from claiming to be lottery winners and cashing out. A brief sketch of the lottery process is provided as follows. After a task has been completed, the payer initiates a new lottery to pay out the participating providers. The payer creates a lottery description L that contains a unique lottery identifier and calculates its hash h(L). The hash is written to the Ethereum contract storage. The payer also announces the lottery description L to the Golem network, so that every participating node can verify that the payment has the correct value and check that h(L) is indeed written to the Ethereum storage. The winner of a lottery payout can be uniquely determined by cross-referencing a given description L and a random value R that is unknown to all parties except the lottery smart contract. After a certain amount of time, if the reward has not been claimed, the smart contract computes the winning provider's address (given the L and R) and transfers the reward from the contract's associated storage to the winner. The hash of the lottery h(L) is also removed from the contract storage and a new lottery payout cycle can now begin. The nanopayment-based lottery payment system is illustrated in Figure 7-6.

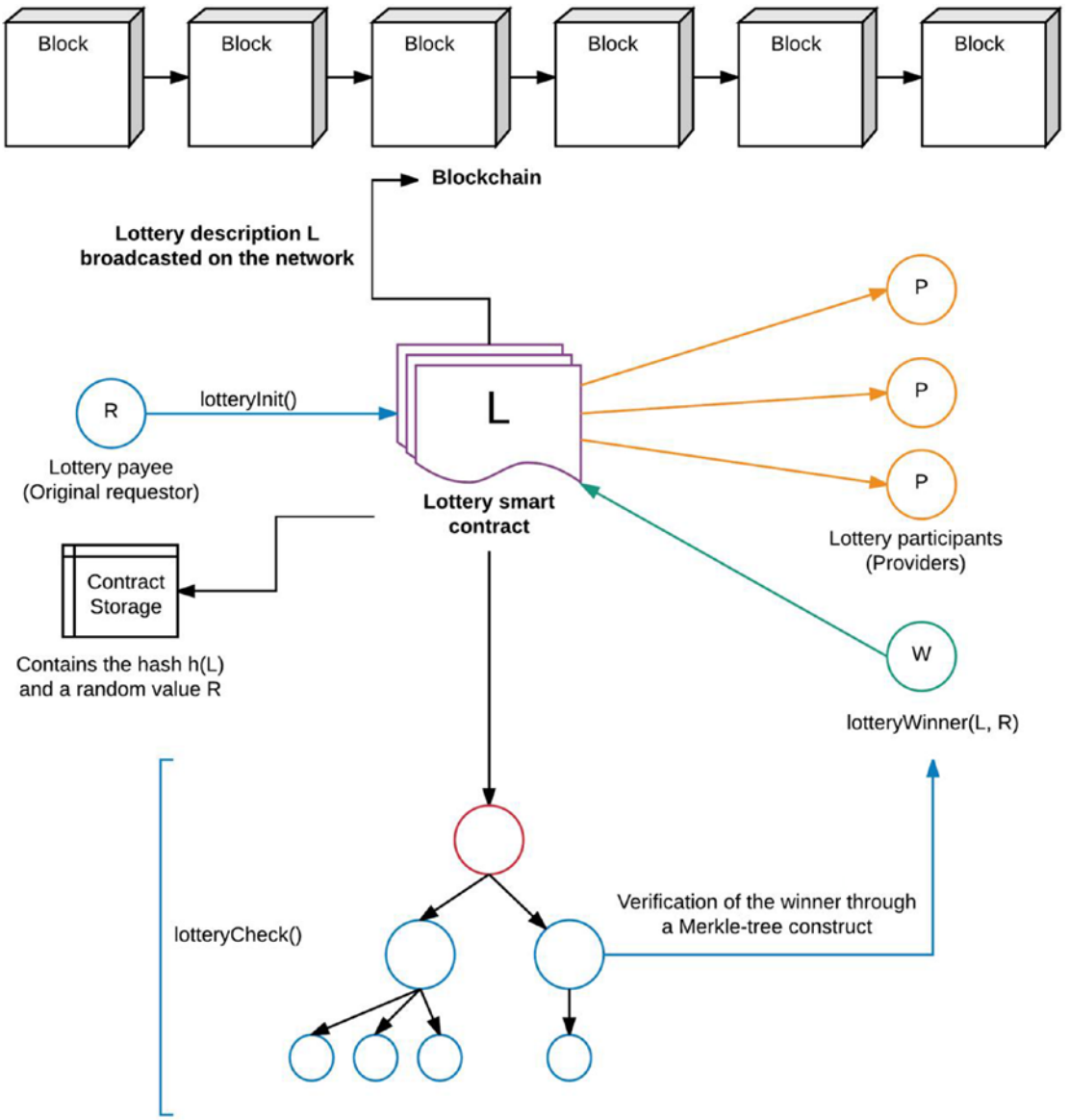


Figure 7-6. *Nanopayment-based lottery payment system*

On a macroscale, Andrzej Regulski wrote a post describing some of the economic principles behind GNT and long-term appreciation in value of the token enabled by this transaction framework. The most pertinent items from his post are quoted here:

> *GNT will be necessary to interact with the Golem network. At first, its sole role is to enable the transfer of value from requestors to providers, and to software developers. Later on, the Transaction Framework will make it possible to assign additional attributes to the token, so that, for example, it is required to store deposits in GNT.*
>
> *The number of GNT (i.e., the supply) is going to be indefinitely fixed at the level created during the Golem crowdfunding. No GNT is going to be created afterwards.*
>
> *A depreciation or appreciation of the token is neutral to the operations of the network, because users are free to set any ask/bid prices for compute resources in Golem, thus accommodating any fluctuations in GNT value.*
>
> *The constant amount of tokens will have to accommodate a growing number of transactions, hence increasing the demand for GNT. With a reasonable assumption that the velocity of the token (i.e., the number of transactions per unit of the token in a specific period) is constant over time, this conclusion can be drawn from the quantity theory of money. This, in turn, means that the overall success and growth of the Golem network implies a long-run appreciation of the GNT.*

This payment framework can also be used as a fallback mechanism for conflict resolution. If a task challenge remains open after going through the traditional Golem mechanics, we need a method for hard resolution (as we saw in the ECM section). Here, we can use a TrueBit-style "trial" for final resolution. TrueBit is a smart-contract-based dispute resolution layer built for Ethereum. It integrates as an add-on on top of the existing architecture for Ethereum projects such as Golem. The design principle for TrueBit is to rely on the only trusted resource in the network to resolve a dispute: the miners. A hard resolution using TrueBit relies on a verification subroutine involving limited-resource verifiers known as judges.

In TrueBit's verification game, there are three main players: a solver, a challenger, and judges. The solver presents a solution for a task, a challenger contests the presented answer, and the judge decides on whether the challenger or solver is correct. The purpose of using the verification subroutine is to reduce the complexity of on-chain computation. This is done as the game proceeds in rounds, where each round narrows down the scope of computation until only a trivial step remains. This last step is executed on-chain by a smart contract (judge) who issues the final verdict on which party is correct. Note that the judges in this scheme are constrained in terms of computational power. Therefore, only very simplistic computations are ultimately carried out on-chain for the judge to make a ruling. At the end of this game, if the Solver was in fact cheating, it will be discovered and punished. If not, then the Challenger will pay for the resources consumed by the false alarm. We provide an outline sketch of the verification game as follows. A visual guide to the verification game is provided in Figure 7-7.

■ **Note** The verification game works iteratively, with a reward structure benefiting the verifiers (challengers) who diligently search for errors. Accurate detection and eventual verification of errors results in challengers being rewarded with a substantial jackpot.

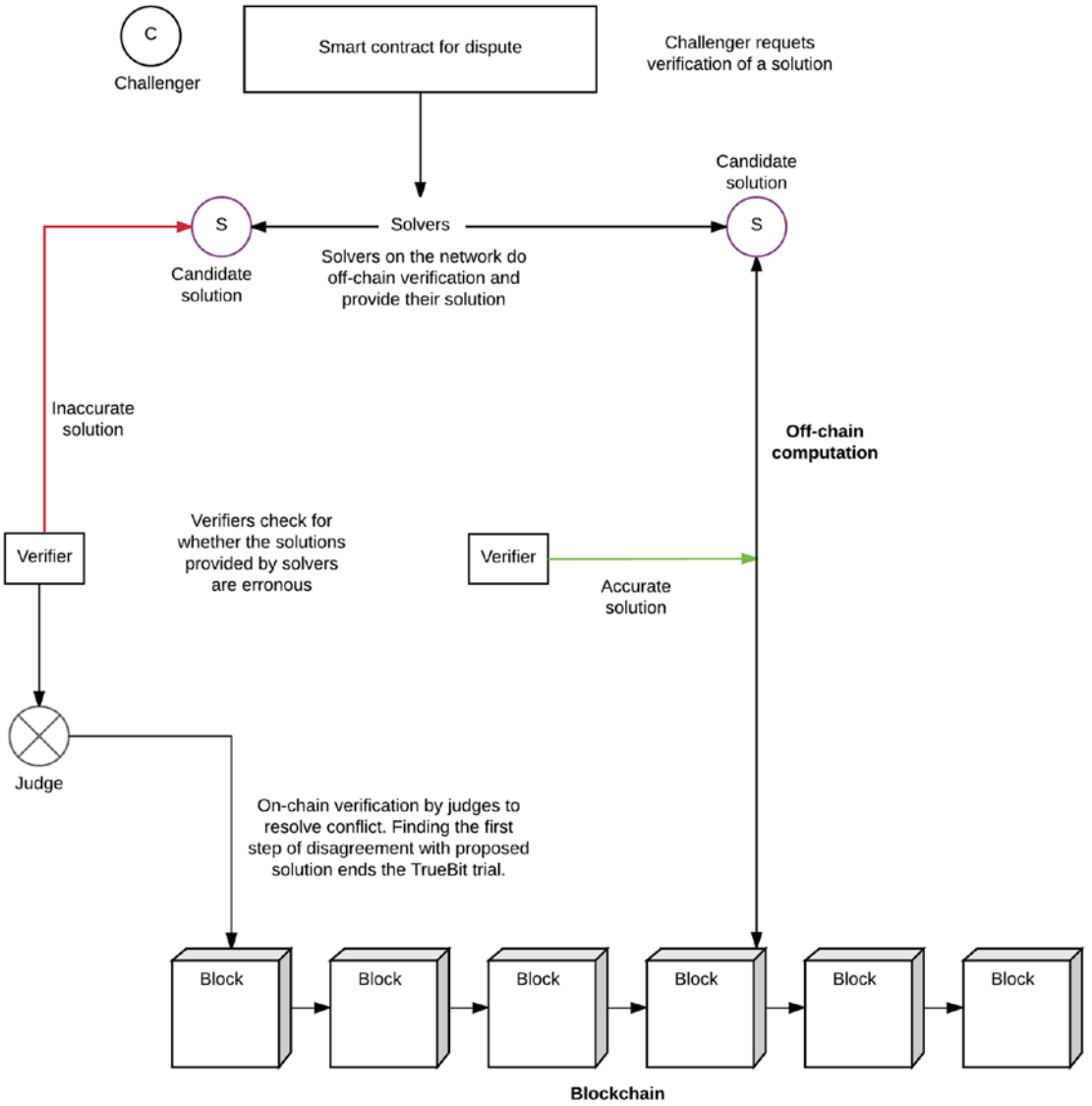


Figure 7-7. *Verification game*

To begin the trial, a challenger requests verification of a solution. The solvers get to work and provide their solutions. Verifiers get paid to check whether the solvers are providing accurate solutions for off-chain processing. If there is still a dispute, judges will perform on-chain processing looking for the first disagreement with the proposed solution. In the end, either the cheating solver is penalized, or the challenger will pay for the resources consumed by the false alarm.

TrueBit is an advanced on-chain execution mechanism compared to the on-chain verification implemented in ECM. It allows the ability to verify computations at a lower cost than running the full instruction because only one step is executed on-chain. Using TrueBit as a dispute resolution layer for off-chain computations, smart contracts can enable third-party programs to execute well-documented routines in a trustless manner. Even for advanced machine learning applications such as deep learning that require

terabytes of data, as long as the root hash of training data is encoded in the smart contract, it can be used to verify the integrity of off-chain computations. The reason TrueBit works for large data sets is simply due to the Merkle roots mapping the state of the network at a given time t, and a challenger's ability to perform binary search. For a project like Golem aiming to achieve HPC toward a variety of use cases, TrueBit can enable large data sets to be used off-chain with an on-chain guarantee of accurate outputs.

Supercomputing Organized by Network Mining

SONM is a decentralized implementation of the fog computing concept using a blockchain. To understand how SONM works within the framework of fog computing, we first need to define two networking concepts: IoT and IoE. The European Commission defines the IoT architecture as a pervasive network of objects that have IP addresses and can transfer data over the network. IoT lends itself to the broader concept of Internet of Everything (IoE), which provides a seamless communication bus and contextual services between objects in the real world and the virtual world. IoE is defined by Cisco as:

> *The networked connection of people, process, data, and things. The benefit of IoE is derived from the compound impact of connecting people, process, data, and things, and the value this increased connectedness creates as "everything" comes online.*

Whereas IoT traditionally refers to devices connected to each other or cloud services, IoE is an umbrella term for a heavier focus on people as an essential component attached to the business logic for generating revenue. From an engineering standpoint, IoT is the messaging layer focusing on device-to-device communication and IoE is the monetization layer allowing startups to capitalize on the interaction between people and their devices. A staggering amount of data is generated from devices connected in an IoT network. Transferring these data to the cloud for processing requires enormous network bandwidth, and there are delays between when the data are generated and processing these data and receiving the results (from hours to days). A major limitation of IoT technology is that transfer delay period. Recently, a growing concern has been the loss of value on actionable data due to the transfer and processing stages: By the time we receive those processed results, it becomes too late to act on those data.

One solution presented by Ginny Nichols from Cisco is called fog computing. Fog computing reduces the transfer delay by shifting the processing stage to lower levels of the network. Instead of processing a task by offloading it to a centralized cloud, fog computing pushes high-priority tasks to be processed by a node closest to the device actually generating the data. For a device to participate in the fog, it must be capable of processing tasks, local storage, and have some network connectivity. The concept of fog computing is a metaphor for the fact that fog forms close to the ground. Therefore, fog computing extends the cloud closer to the devices (ground) that produce IoT data to enable faster action. In fog computing, data processing is said to be concentrated at the edge (closer to the devices generating data) rather than existing centrally in the cloud. This allows us to minimize latency and enable faster response to time-sensitive tasks or reduce the time within which an action can be taken on data. SONM makes this layer of fog computing available to a decentralized network of participants for processing computationally intensive tasks.

■ **Note** The need for faster response time to data (or business analytics) collected by large enterprises stimulated the development of real-time computational processing tools such as Apache Spark, Storm, and Hadoop. These tools allow for preprocessing of data as they are being collected and transferred. In a similar manner, fog computing seems to be an extension of IoT developed in response to the need for rapid response.

SONM is built on top of Yandex.Cocaine (Configurable Omnipotent Custom Applications Integrated Network Engine). Cocaine is an open-source PaaS stack for creating custom cloud hosting engines for applications similar to Heroku. SONM has a complex architecture designed to resemble the world computer model of Ethereum. The SONM team designed this world computer with components that are parallel to a traditional personal computer, with one major difference: The components are connected to a decentralized network.

- *CPU/processor (load balancer)*: In SONM's architecture, the processor (hub) serves as a load balancer and task scheduler. The entire network can be represented as a chain of hub nodes (or hubs) that distribute tasks, gather results from a computing fog, pay miners for services, and provide status updates on the overall health of the network. Each hub node is analogous to a processor's core, and the number of cores (or hub nodes) can be extended or reduced on the network as necessary. In a personal computer, the cores are locally accessible to the processor, however, in SONM, the cores are decentralized by nature. The hubs provide support to the whole network for coordinating the execution of tasks. More specifically, hubs provide the ability to process and parallelize high-load computations on the fog computing cloud.
- *BIOS (blockchain)*: For SONM, the BIOS is the Ethereum blockchain. Ethereum offers a reliable backbone for network consensus and payment mechanisms that SONM can inherit. However, the base Ethereum implementation lacks a load balancer and high gas costs have inspired alternatives on top of the blockchain such as SONM.
- *Graphics processing unit (GPU)*: The GPU for SONM is the fog computing cloud. More specifically, this fog is comprised of the miners in the SONM network that are making computational resources available to buyers.
- *Connected peripherals*: The buyers in SONM are equivalent to peripheral devices. They connect to the network and pay for any resources they use. The requests are broadcast on the blockchain and miners select what they want to execute on their machines. Then, the load balancer dictates the execution of tasks.
- *Hard disk*: Storage in SONM will be implemented using well-established decentralized storage solutions such as IPFS or Storj.
- *Serial bus*: This communication module is used for message passing and data transfer within the network between the sender (a node) and working machines. This serial bus is based on Ethereum Whisper, and enables broadcast and listening functions for messages on the network.
- *Plug-ins circuit board*: A plug-ins board can be thought of as an expansion pack for the network. It allows SONM to expand its processing capabilities by seamlessly integrating compatible networks and computation farms.

Figure 7-8 provides a visual representation of the world computer's components that just discussed.

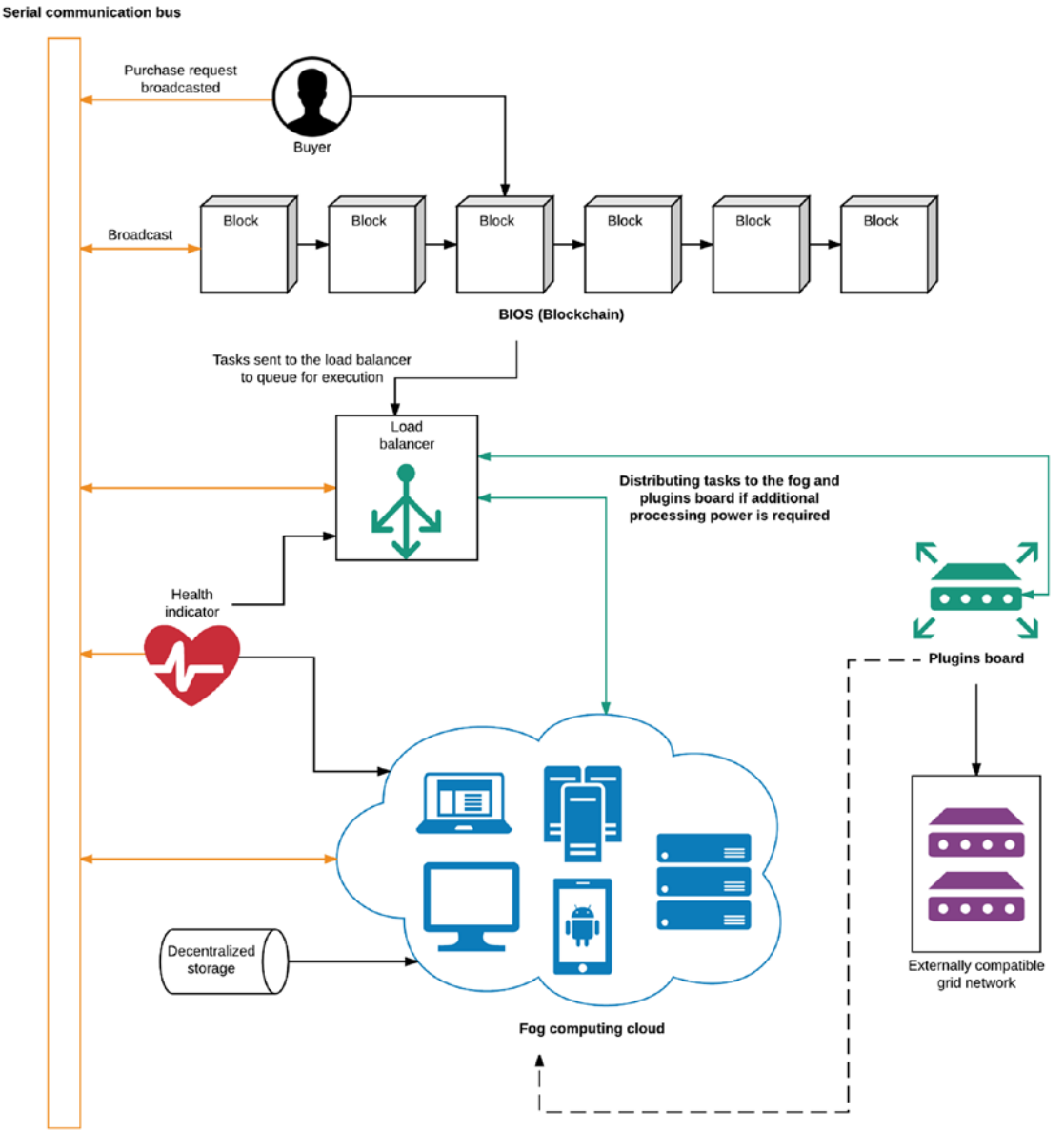


Figure 7-8. *World computer implemented in SONM*

The communication bus forms the backbone of this architecture, connecting at every level to the components of the world computer. The workflow of a request through SONM begins with a buyer requesting computational resources. This request is broadcast on the blockchain and the serial bus so that other components can verify if a particular request is legitimate. The miners accept the task, and the load balancer distributes the task to be executed in the fog computing cloud. If necessary, the load balancer can also assign a task to run on an application-specific grid network compatible with SONM through the plug-ins board. The fog cloud also has local storage in the form of a decentralized service such as Storj or IPFS that is used as necessary. Finally, the load balancer also has two more important roles, to follow up with the task results and overall network health. The load balancer (hub) collects the results of a task after it has been completed and sends it back to the buyer. A health indicator reports on the status of every component in the world computer to the load balancer and the network to make better decisions regarding task assignment and execution.

The smart contract system in SONM is a set of protocols (smart contracts) for blockchain governance and management of decentralized applications implemented on the network. This system of smart contracts maintains the integrity of the network and provides several countermeasures against malicious entities that we discuss later. Furthermore, SONM uses formal definitions and rigorous proofs to demonstrate the security of communication (cryptographically secure message passing) between buyers and providers. The interactions between various parties on the network are modeled using a high-level mathematical framework called pi-calculus. This framework is a formal language for describing secure and concurrent interactions within a multiparty system over well-defined channels. The governance prototype in SONM includes a DAO, a court contract, a network-wide registry of hubs, and a factory contract to deploy new applications to the SONM network (e.g., the whitelist, hub factory, and hub wallet). This model is extended to include some new elements (deployed by the factory) necessary for management of DApps that run on the network. Currently, the process of buying computational resources appears very fragmented on the network. However, in future releases, buyers will complete a prepurchase computing form to indicate their hardware selection, the DApps they want to use, and any special execution parameters. Let's look at the complete set of smart contracts used in the current SONM blockchain government in more depth.

- *SONM token*: This is the default currency of the network. The token is used to reward miners for providing computational resources and for user transactions within the network.
- *DAO*: This is the regulatory body of SONM that grants users voting rights and administrative actions to manage decentralized applications. DApps on the network are required to pay taxes to be included in the DAO and gain access to the court. The DAO also has executive privileges to perform actions such as blacklisting a hub, releasing and locking funds to ensure proper payout for miners, and suspending or banning a hub.

■ **Note** The court will be implemented as a dispute resolution smart contract accessible through the DAO. It will offer protection against unfair buyers or providers in the market, along with result verification in case of challenges toward a solution.

- *Hub factory*: A hub has two extensions in the smart contract system, a factory and a wallet. The wallet can only be created by a hub wallet factory. The factory creates a new wallet contract and then registers this contract in the whitelist.
- *Hub wallet*: The hub wallet is a contract that receives payments from buyers and facilitates the payout of tokens to miners for services. The wallet is created by a factory and it can exist in one of four possible states: Created, Registered, Idle, or Suspected/Punished. Initially, all wallets exist in the Created state and have a fixed amount of frozen funds. In the Created state, a contract can be registered on the whitelist. This contract now switches to the Registered state, and has access to advanced functions such as transfer and payday. Now this wallet can begin to pay out miners (but not itself) within a specified amount of time, called the payout period. After the end of the payout period, the wallet can transfer the remaining money to the owner's wallet using the payday function. If a connected hub is caught performing malicious behavior, the DAO can blacklist the hub and seize the frozen funds. The DAO can further decide to change the state of the wallet to Suspected or Punished depending on the case.
- *Whitelist*: The whitelist is a registry-type contract that contains detailed information about hubs across the network along with their status. All wallets created by the factory are registered in this whitelist contract. Initially, this whitelist serves as a registry for trusted hubs verified by SONM developers to be safe and secure. Eventually, this functionality will be expanded to open ratings for new and emerging hubs.
- *RegApp*: This is a simple web app made with `React.js` used for hub-node registration. This application adds a node to the whitelist contract. It should be noted that the RegApp is language agnostic and only needs a web interface to register the node. In this instance, `React.js` was used, but other web programming languages can work just as well.
- *PayOut app*: A payment application that runs on the SONM network to pay and process the tokens given to miners for their services. This application is an example of the types of DApps that a factory contract would deploy.

Figure 7-9 provides a visual companion to the smart contract system just discussed. Anthony Akentiev from Chain Cloud reviewed the SONM whitepaper and provided some comments regarding the crowdfunding efforts using SOMN token to fund the development of this smart contracts architecture in future releases:

> *The crowdfunding is divided into 2 stages—the presale and the ICO. 19% and 20% of all funds are sent to the team as a reward. Development and marketing will require equal amounts of money: 34% for the development and 32% for marketing after the presale; 30% for the development and 33% for marketing after the ICO. It looks good and fair.*
>
> *Conclusion—45/60*
>
> *One of the best Whitepapers that I have read. Big project. Big goals. The presale and ICO are gonna be big.*

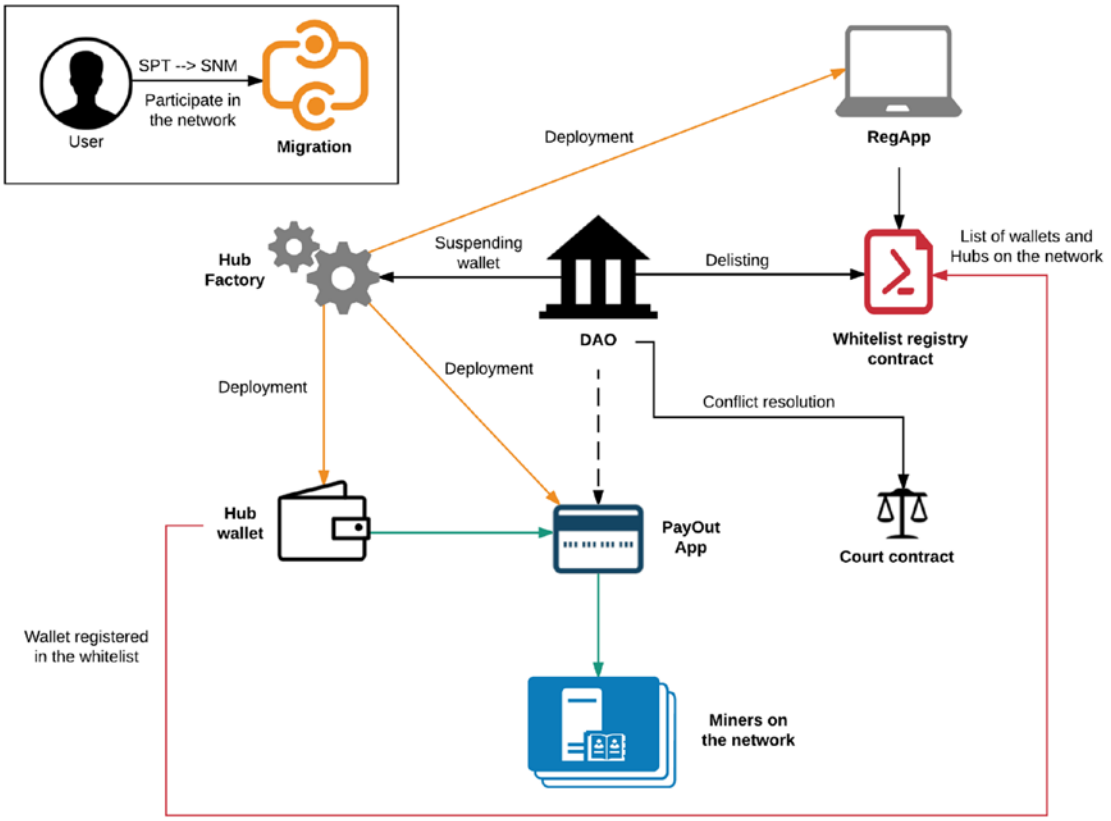


Figure 7-9. *Overview of the smart contract system in SONM*

Recall that the smart contract system in SONM is further made from several contracts working together. The first of these contracts is a migration function that allows users who purchased the SONM presale token (SPT, during the presale that Akentiev mentioned) to exchange their token for the SONM token (SNM) and participate in the network. Next is the hub factory contract used in SONM for deployment of applications. In the current implementation, the hub factory creates the hub wallet used to compensate miners for their computational resources. The factory also deploys a network-wide whitelist contract that serves as a registry of hubs active on the network and wallets created in each hub. Finally, the factory deploys a payout application that releases SNM tokens to miners for the provided services. The last smart contract we consider here is the DAO, which provisions several administrative functions to stop malicious entities. These functions include delisting a hub (from the whitelist contract) if caught in fraud, freezing funds by suspending a hub wallet, or conflict resolution in case of challenges using the court.

Buyer–Hub–Miner Interactions

Posting a task on the SONM network requesting computational resources requires a fair amount of communication between the buyer and the miner who will execute the task. For instance, before processing can begin, there are several verification interactions that occur between a miner, the assigning hub, and the buyer. Once the buyer is verified, a hub will assign the task to a miner's fog computing cloud and ensure the miner is paid for their services. This collective set of interactions is defined here as client–hub–miner

communication. This interchange can be best demonstrated visually in Figures 7-10 and 7-11. Figure 7-10 explains the miner–hub interactions when a computation begins and Figure 7-11 provides the full picture by introducing the client and miner payout.

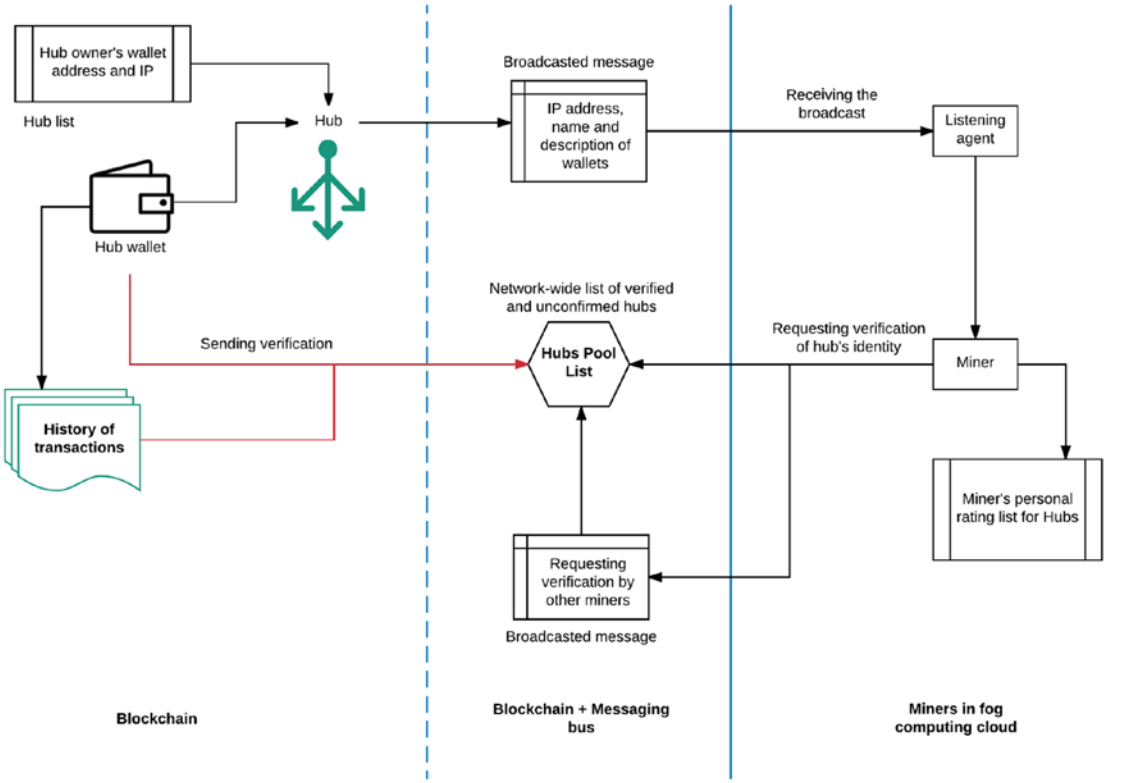


Figure 7-10. *Miner–hub verification*

A hub broadcasts information about an active hub wallet, and the hub owner's IP address to the blockchain. This information is gathered by a listening agent and then dispatched to a miner. Before executing any tasks, a miner will go through a series of steps to validate the assigning hub. This process ensures that the hub can be trusted and that there are sufficient funds to pay for the computation in the hub wallet. A miner requests more information on the assigning hub through a global list called the hubs pool list. This network-wide list contains all the verified and unconfirmed hubs. To verify itself, a hub wallet sends its address on the blockchain and a history of past transactions as proof of authentication. The miner accepts the proof and can now begin the computation. It must be noted that all components of this validation process including hub broadcasts, verification requests, wallet information, and hubs pool list use the same infrastructure: the blockchain. The requests and broadcasts are shared to the rest of the network using the messaging bus built with Whisper on top of the blockchain. The dashed blue line in Figure 7-10 indicates a continuation of processes happening on the blockchain with notifications moving to the messaging bus. In addition to the pool list, the miner can also request other miners to independently verify the assigning hub's credentials. Over time, the miner builds a personal rating list for hubs across the network that have assigned tasks. This list can be used to automate task acceptance and skip the verification steps for a highly trusted hub.

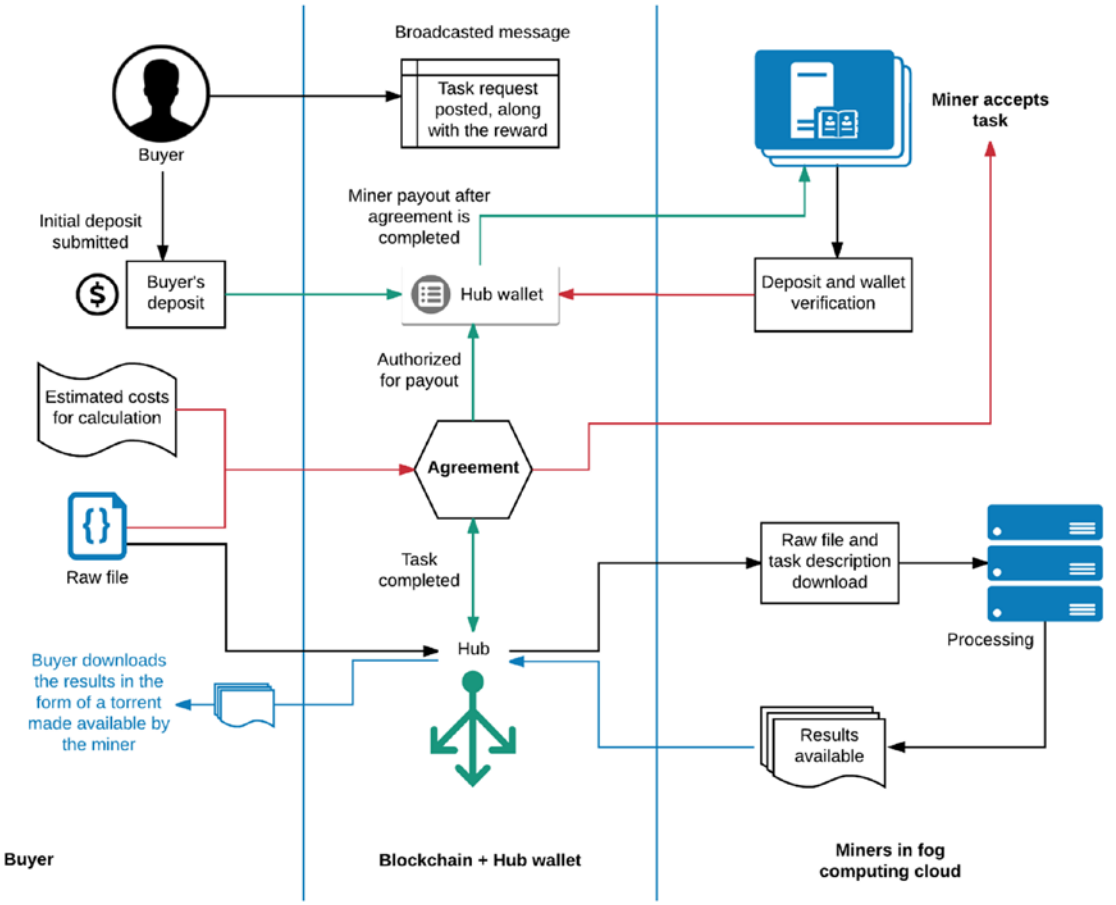


Figure 7-11. *Buyer–hub–miner interactions*

After the initial verification of the assigning hub has been completed, the buyer puts forth an initial deposit. This deposit is a rough estimate of the computation costs, and it is stored in the hub wallet. The miner verifies this deposit and the buyer signs an agreement on the blockchain with the estimated costs along with a hash of the raw file. This agreement is sent to the miner as the computational task is assigned by a hub. The raw file is made available to the miner for download through a torrent. After the file has been downloaded, the task description along with any special parameters are applied and the processing begins. Once the task is completed, the results are made available from the fog computing cloud, and the hub collects these results. This hub now makes the results available for the buyer to download through a separate torrent. At this point, the agreement has been completed and funds from the hub wallet are released to the miner for offering their computational resources. In future releases, there will be proper locking mechanisms to account for over- or underestimation of the costs. Until the buyer adjusts for any errors in the estimates, the results will not be released by the hub.

Superglobal Operation System for Network Architecture

So how does a buyer know whether a miner satisfies the required dependencies to compile and run an application that the buyer requested? How does a buyer know what applications are compatible with a miner's setup? These questions are handled by the operating system for SONM. A world computer is not complete without an operating system (OS) to power it, and the OS for SONM is called the Superglobal Operation System for Network Architecture (SOSNA).

■ **Note** In this chapter, we have discussed SONM only in the context of buyers and providers for HPC services, however, the first use cases for early versions of SONM focuses on running a Quake server in a completely decentralized manner. The roadmap includes the message passing protocols and the payout app next.

The problem of dependency management and compatibility for compilers will be handled by using containers on host machines. A container is simply a minimal virtual machine that can load any specified configuration, from installing new libraries to standard tools such as the GNU compiler toolchain. The container runs in a minimal isolated environment on the host system. This is done to prevent any escalation of privilege attacks by malicious programs running within the container. SOSNA can be divided into three main components: the blockchain layer comprised of the smart contract system, a grid core, and a layer of consumer-facing applications. We already discussed the blockchain layer and API (enabled by the smart contracts) in detail. Let's shift our focus to the grid core. In SONM, any grid-compatible PaaS stack can be plugged in to work with the network such as BOINC (Berkeley Open Infrastructure for Network Computing) or Yandex.Cocaine.

A simplified implementation of grid architecture involves two units: multiple worker/slave modules and a master module. These two units form a computing cloud, and a fog computing cloud extends this idea to hundreds of instances deployed across all types of machines. What defines a grid network as such is that a master machine manages workers distributed across distance to different geographic locations. The master essentially behaves like a hub, as it manages execution of applications on miners' machines, balances load, schedules tasks, and collects the results. More generally, we can suggest that a master module works like a traditional miner pool.

Any application executing inside a container is referred to as a service. Technically, all services in SONM are RPC-based and accept a specific set of messages to execute directives. Each service has a list of functions that can be referenced by sending messages to that service after a connection has been established. This set of functions is called a service protocol. A protocol can be dynamically obtained from a service once it becomes active by resolving the service name using a locator. The use of a locator can be thought of in analogy to hostname resolution using DNS. We return to the locator service shortly. On startup, a node loads the settings specified in a configuration file, including all the services to start on this particular node. The services lack the ability to communicate code over the network, so they can only receive messages in this state. To allow access to the network and RPC-compatible communication, the node will start a special service called the locator. Every other service running in the container becomes attached to the locator for receiving and sending messages. Once the locator service is activated, it binds to a public endpoint in the network and broadcasts the connection to the blockchain. What steps does a buyer need to perform to access a particular service for their task? Using the locator service, a buyer needs to go through the following five steps:

1. Connect to locator endpoint on a public port.
2. Send a resolve request with the name of a service to the locator.
3. Receive a message back from the locator with information about the endpoint container, the protocol description, and function calls.

4. Receive a confirmation message indicating that the request for information was completed.

5. Request for a specific miner with the endpoint information that matches the locator message and call the required service for task execution and processing.

A visual companion to the components of SOSNA is provided in Figure 7-12.

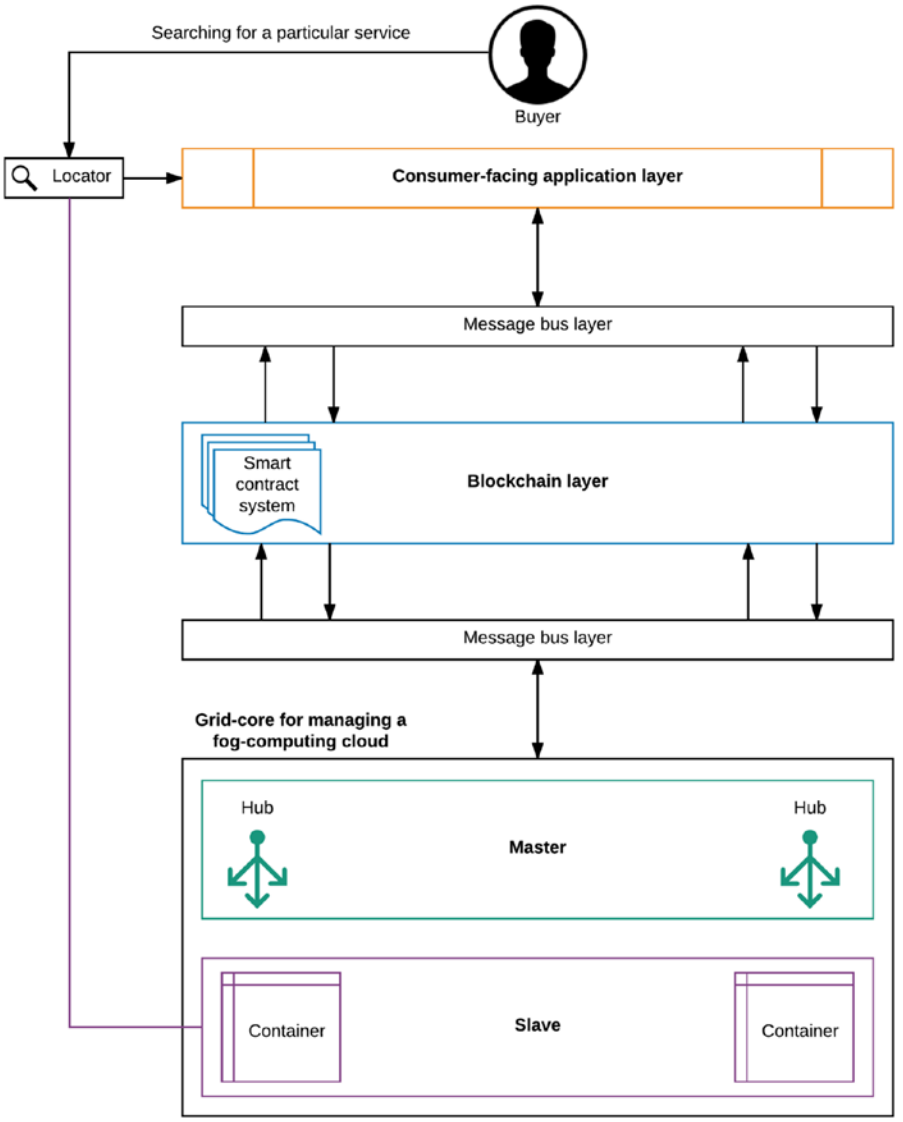


Figure 7-12. *Overview of SOSNA architecture*

We start at the blockchain layer, which provides the consensus framework to the network, along with the smart contract system for governance. Next is the messaging bus layer that serves as a two-way communication channel between the various components of the world computer and users in the network. This layer also exists on both sides, allowing the blockchain to communicate to the consumer-facing applications and the grid core to coordinate task execution. On the very top, we have a consumer application layer where buyers can access the network and post requests for services. The RegApp is an early example of consumer applications that buyers would interact with in the future. Eventually, the consumer layer would split into more verticals with features such as trusted runtime skipping the verification stages from users with a high reputation in the network. On the opposite end, at the very bottom we have the grid core. This is the heart of SONM where the fog cloud computing takes place. The grid is essentially a parallel implementation of several hundred master and slave modules. The slave modules run containers where services execute, and these services are connected to a locator. The locator runs on a public port accessible to the buyer through the consumer-apps layer.

iEx.ec

We conclude this chapter with a short discussion of the iEx.ec platform, which is also a distributed HPC token using a domain-specific blockchain. iEx.ec is built on a mature set of research technologies called XtremWeb developed for desktop grid computing by INRIA. So what is a desktop grid? The idea behind desktop grid computing is to collect the computation resources of idle machines connected to the network to execute intensive parallel applications, at a very small cost (or free, in the case of volunteer desktop grid) compared to a traditional supercomputer. XtremWeb-HEP is a decentralized version of its predecessor, designed to execute tasks that require intensive computing resources in a marketplace of miners. This open-source desktop grid stack belongs to the Cycle Stealing family that uses idle resources on machines connected to the network for executing data-intensive applications. XtremWeb-HEP implements most of the features necessary in an HPC token that we have previously seen: fault tolerance, hybrid public–private infrastructure, deployment of virtual images (containers), load balancing, and a payment mechanism for the miners. The DApps running on the network will rely on iEx.ec architecture to automatically search for all the computing resources needed to run the app, provision resources, and release funds to the appropriate parties. Using a well-established and well-tested software stack offers three main advantages:

- *Resilience:* If one computation node fails, the task can be reassigned to other working nodes and continue with minimal downtime.
- *Efficiency:* The base performance of applications across the network is high, despite hardware configuration changes that occur for workers from one node to another.
- *Pluggable nodes:* New nodes can be added to the network without any special configuration, and nodes become integrated after a quick setup.

In iEx.ec, any actions that happen externally or off-chain are referred to as a contribution. For instance, providing a data set, transferring files, and performing a computation are all actions that would require token transactions between the relevant parties. How do we know whether the action actually took place correctly? How can a particular transaction or set of transactions be associated with a particular action that was performed off-chain? A new protocol is needed to verify that a contribution took place and which transactions correspond to it. iEx.ec has proposed a proof-of-contribution as a new consensus mechanism for off-chain contributions. This new mechanism also plays a role in enabling an enterprise feature in iEx.ec called a service-level agreement (SLA). Using an SLA allows for resource utilization tracking and trusted computing resource orders from the customer to a provider. The iEx.ec team predicts that content distribution to parties on the network will be a tremendously important function of DApps using the iEx.ec blockchain. The buyers will have access to complex data sets on the blockchain through a smart contract, and a simple payment structure for running their application along with access to the data set. Using proof-of-contribution, iEx.ec

can guarantee that the content providers are actually giving access to data sets, and that the file was accessible during the processing. For payment purposes, the duration of access can also be recorded to protect the buyer against overcharging. In addition, iEx.ec will have several countermeasures against malicious entities that claim file transfers failed to reduce data charges.

The first few releases of iEx.ec will focus on consolidating the core features and a financial trading use case that requires HPC. This service, called eFast, will employ sophisticated computational prediction methods to allow small investors to make better trading decisions. The objective is to create diverse portfolios using cluster analysis of different stocks, but the computational complexity of such an analysis is so immense that only large financial institutions can afford it. A decentralized service like iEx.ec can reduce the processing cost to one tenth of what it would traditionally cost on a computation farm.

■ **Note** Golem and iEx.ec share similar goals for product development, but they differ in business approach. Golem aims to build the core infrastructure of the network and attract HPC and cloud customers. On the other hand, iEx.ec first focuses on building DApps that will run on the network to attract regular cloud customers that have a need for those applications at a cheaper rate.

How do tasks get assigned and executed on iEx.ec? There are two algorithms that manage this process, a matchmaking algorithm for execution and a scheduler for assignment. Let's briefly talk about both of them. A matchmaking algorithm is used in distributed systems to match pairs of resource providers and potential buyers according to the description of resources provided by the buyer. Recall that in SONM, this task was taken over by a load balancer; however, in iEx.ec, an algorithm performs resource provisioning. Eventually, the developers propose to store smart contracts on the blockchain that describe the availability of computational resources and the resources required to run a task. This can include information such as available RAM, processor or CPU, disk space, and the type of GPU. For a host machine, the tasks would be deployed to a virtual machine instance or a hypervisor, and the buyer would specify the runtime instructions. The matchmaking algorithm can implement more intricate policies beyond a simple one-to-one match and iEx.ec can monetize the more complex matches. There are several design choices for the matchmaking algorithm, but iEx.ec will be using a language called ClassAd, which has been well tested in the scientific literature. The second algorithm is the scheduler, which is critical to any distributed computing system. The performance and scalability of a network depends on the effectiveness of its scheduler. The design challenge for iEx.ec is to create a scheduler enabled with multiple-criteria decision analysis that can select optimal computing resources for each task and schedule it appropriately to fit the buyer's criteria. A multicriteria might fit a customer's needs based on the cost vs. performance benchmarks. One customer might want to minimize the price of computation if it takes longer, whereas another might want to minimize the time required. These are the types of scenarios that will be handled by a scheduler in iEx.ec.

A versatile schedule and efficient matchmaking algorithm allow for very interesting upgrades to the core blockchain technology in the context of HPC. One remarkable example is the ability to use domain-specific blockchains (or sidechains). A sidechain is an adaptable blockchain designed to meet specific infrastructure requirements for running applications that would underperform in a generalized environment. The goal of such a blockchain is to maximize performance and minimize any time delays between execution. Normally, an application would go through the standard blockchain; however, in the case that a large amount of tasks are submitted, some of the tasks can be offloaded to a sidechain for tracking. The sidechain will track the amount of time spent on host machines and report the information back to the network. This offloading can reduce the traffic on the generalized blockchain. An advanced system of smart contracts would be developed to enable this switch. Figure 7-13 provides a visual companion to the iEx.ec architecture.

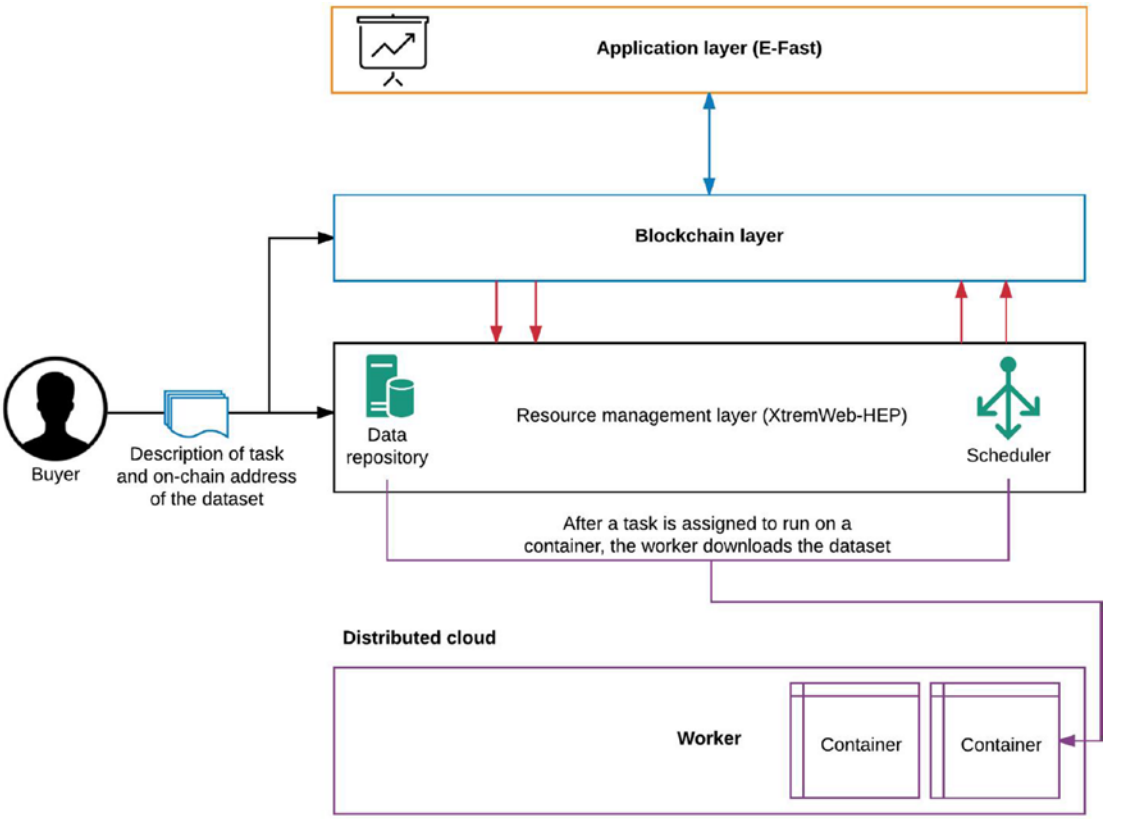


Figure 7-13. *Overview of iEx.ec architecture*

The top layer is a consumer-facing application, E-Fast. It is the first use case of iEx.ec that provides the infrastructure to run sophisticated financial prediction algorithms in a massively distributed setting for a fraction of the cost. Underneath is the Ethereum blockchain layer that provides consensus for the network and task-tracking for reimbursement of miners. The blockchain also acts as a platform for the smart contract system that will eventually run in the network. It is important to note that the blockchain layer does not interact with every level of iEx.ec; the distributed cloud level is managed more directly through a middleware XtremWeb-HEP. This is the resource management layer comprised of a task scheduler and a verified data repository. The scheduler coordinates between the blockchain and assigning tasks to workers on host machines. The repository contains large data sets from various fields such as finance and genomics that are made accessible to applications that buyers want to execute. The buyer connects to the resource management layer and provides a description of the task along with the source of data. This information is also recorded on the blockchain. Both components of the middleware connect to the deployed workers and use the description and execute the task. The final results are collected by the middleware and made available to the buyer after a payment is recorded on the blockchain. The last layer is a distributed cloud of miners. Notice here that we don't need a master module because the middleware performs that function. The tasks are executed in containers or virtual machines and the results are reported back to the resource layer. This reduces the overall complexity of the network, as the only code running on host machines are the containers.

Summary

In this chapter, we looked at Ethereum tokens from the perspective of HPC applications. We first introduced the market utility and rationale behind introducing tokens as apps on Ethereum. Then, we discussed the prototype HPC token called ECM. This token performs all the basic functions that would be needed in a computational market where customers can purchase computational power from clusters run by dedicated buyers. The token allows for dispute resolution and on-chain verification in a transparent computational power market. We then delved into the more complex computational tokens, such as Golem and SONM. We described both in heavy detail, covering the major technical advancements and how those two tokens differ from each other. Finally, we concluded by covering iEx.ec, which is built on distributed cloud computing software that has been tested for years. The iEx.ec team implemented a decentralized version of Xtreme-Web HPC to perform the same task as Golem and SOMN and enable a computational market.

References

The main references used to prepare this chapter were the ECM GitHub documentation, Golem whitepaper, SONM whitepaper, iEx.ex whitepaper and Fred Wilson's blog post on tokens.

CHAPTER 8

Blockchain in Science

Evidence-based clinical sciences are currently suffering from a paralyzing reproducibility crisis. From clinical psychiology to cancer biology, recent metaresearch indicates a rise in researchers failing to replicate studies published by their peers. This problem is not just limited to benchwork that happens in a lab; it also plagues translational research where the transformation from bench to bedside happens. Treatments, tests, and technologies are converted from simple lab experiments to government-approved devices and assays that affect hundreds of lives. Therefore, replicability is crucial to converting scientific breakthroughs into pragmatic remedies.

The primary role of blockchain technology in the emerging landscape of open-access science is increasing the amount of transparency into the processes. To that end, three use cases and applications are presented in this chapter: The first one involves deposition of data in clinical trials, the second one involves a reputation system that can be developed for researchers and institutes committed to open research, and the third one involves the application of supply-chain management for tracking counterfeit drugs. We begin by discussing the current paradigms of the research method and the importance of negative data, and our focus remains limited primarily to the clinical sciences. Then, we talk about traditional metrics and altmetrics systems presently implemented to measure the impact of published research. This allows us to transition into the use cases that supplement traditional metrics and expand them to make open science a fundamental feature. Finally, we end our discussion by looking at ongoing efforts in psychology research to incorporate prediction markets to verify research, and how prediction markets can be created using Augur.

Reproducibility Crisis

Let's begin our discussion with what *reproducibility* means in the context of scientific discourse and investigation. One of the major cornerstones in research is the ability to follow written protocols from a study, using the documented methods to carry out experiments and arrive at the same conclusions given by that study. In other words, a published study can be independently verified by other researchers, and replicated to give the same results. Recent metaresearch in clinical sciences demonstrates that more and more published works are not rigorous enough experimentally to be replicated with ease. Freedman et al. estimated that over 50 percent of preclinical research cannot be translated from animal models to human clinical trials, thereby placing the approximate annual cost of irreproducibility at $28 billion in the United States alone. Consequently, as preclinical findings in animal models are rarely repeated in clinical trials, drug discovery has slowed down and costs have risen dramatically. The economic costs are very debilitating; close to $200 billion is wasted annually because the discovered targets cannot be reproduced. These problems are often inherent to the design of a particular study, or due to very genuine intricacies and complications that arise in experiments on differing cell lines. To understand why, we have to look for answers in metaresearch, which is a branch of investigation that statistically evaluates the claims, results, and experiments performed in a study.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017

V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_8

A multitude of factors in academia are creating a vicious culture of "bad science," as Dr. Ben Goldacre referred to it: a vanishing funding environment, a culture of "publish or perish," and an incredible amount of pressure to get tenured pushes young researchers to follow erroneous methods to get published. In some cases, the slipshod research methods and manipulations have led to fraud and eventual retractions with serious consequences to the researcher.

■ **Note** Retraction Watch (http://retractionwatch.com/) is a blog that recently came about to report on scientific misconduct happening at all levels, from editors working with journals to individual researchers at universities. The blog also tracks papers that have been retracted from journals due to fraudulent data or manipulation of experimental evidence. The interested reader can follow their web site, which posts about 500 to 600 retractions per year.

Academic journals are also partially to blame for this mess, although there are signs of real change and improvement in the works. Over the past years, it has been easier to publish studies with positive findings in journals regarding potential drug targets or effectiveness of a particular drug. Negative findings from experiments, such as a drug target not working even though it was expected to work, have been incredibly difficult to publish, however. At face value, it seems to make sense: Why would someone want to know if an experiment failed? Negative data usually get ignored because of similar reasoning, and from a marketing standpoint, a journal claiming that something did not work as expected is not a highlight that would sell. Let's take a closer look at positive and negative data in the context of how journals have shaped their use and publication.

Positive data simply confirm an initial hypothesis, where a researcher predicted a finding and the data validate it. On the other hand, negative data come from the cases where the expected or desired effect was not observed. If an experiment showed that no difference exists between the null and alternative hypotheses, the data and results will likely end up buried in the pile of unpublished results of the lab group. Figure 8-1 provides a very simplified overview of an erroneous research method for dealing with positive and negative data resulting from the pressures in academia. Replication is sacrificed in pursuit of highly demanding and attractive drug targets, which ultimately do not translate well into clinical trials and lead to more economic waste.

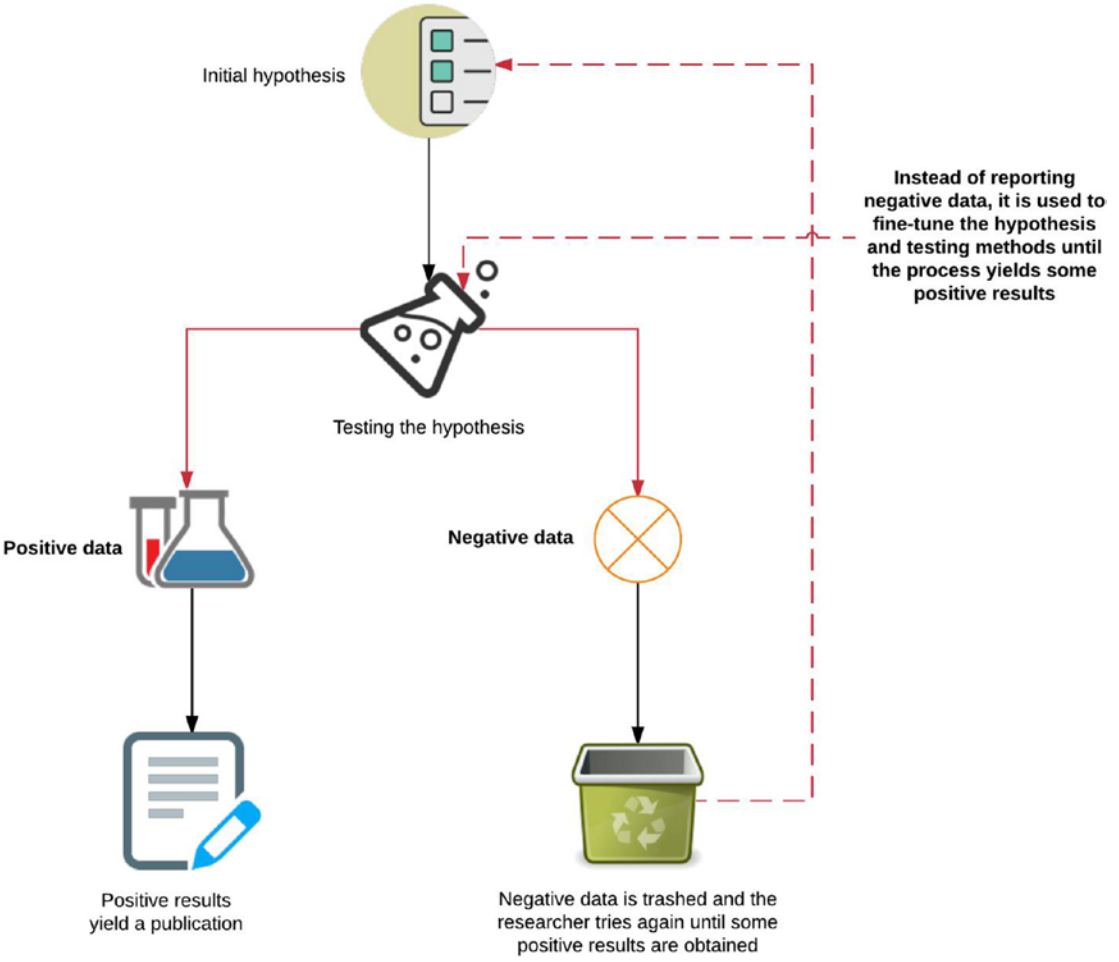

Initial hypothesis
Instead of reporting negative data, it is used to fine-tune the hypothesis and testing methods until the process yields some positive results
Testing the hypothesis
Positive data
Negative data
Positive results yield a publication
Negative data is trashed and the researcher tries again until some positive results are obtained

Figure 8-1. *Overview of positive and negative data in the context of publishing a study*

The flowchart in Figure 8-1 is a simple demonstration of hypothesis testing that leads to publication of "flukes" that are nonreplicable in translational research or scaled clinical trials. Due to the nature of publishing in academic journals, positive data usually imply that you are done. Most researchers will stop here, and not bother to follow up with any appreciable fraction of all the data that were collected or generated while conducting the experiments. This could include potential negative data regarding an avenue of thought that did not work, or information that was omitted due to feedback from reviewers. Once a research paper has been accepted, the authors have no further incentive to release more data or put in more time to clean up and make other results available. This turns out to have some detrimental consequences, which we discuss later in the chapter.

These trends have been observed by journals internationally and they are beginning to take some action. A plethora of new initiatives are raising the standards for the data that can be included in a publication and the design considerations that must be fulfilled to ensure replication. Let's talk about three of those efforts here.

- *Minimum publishing standards*: Print journals have specific requirements for space where only a certain number of pages can be allocated to each section of a research paper. In such scenarios, researchers focus more on making bold claims and showing data that provide evidence for their conjectures. Usually, this comes at expense of shortening the methods section that provides instructions for other researchers to follow to replicate an experiment. Recently, most journals have moved online and space is less of an issue, although supplemental materials are still lacking in quality even when they are made available. BioMed Central has released a checklist of minimum standards that must be met before a paper can be published. The purpose of this checklist is to provide a level of standardization so that researchers can write papers with specific criteria in mind to enhance reproducibility. If all the standards are met, there is a high likelihood that a published study can be replicated to a greater degree.
- *Data Discovery Index*: One of the major problems mentioned earlier was the lack of incentives for researchers to make supplemental data available. The National Institutes of Health (NIH) has sought to create a new measure to credit researchers for uploading additional experimental data called the Data Discovery Index (DDI). This is a citable data repository where researchers can make additional data points available related to their studies. For academic researchers, a huge incentive is to elicit additional citations for their work, which in turn becomes a measure of impact for a published study. By making the database citable, NIH created this new incentive for researchers to dedicate additional time and resources to upload unpublished databases.
- *Reproducibility Project: Cancer Biology*: The Center for Open Science in collaboration with Science Exchange will be looking at high-impact studies in cancer biology from 2010 to 2012 and replicating each one with help from members of Science Exchange. Full-length reports on the attempts at replicating experiments, discovering drug targets, and more will be openly made available along with detailed methods. This project is being done in two phases. The first phase culminates in a registered report that documents standardized protocols to carry out certain experiments. The second phase involves one of the member institutes of Science Exchange conducting the experiment using the registered report and documenting the results. Ultimately, both the reports and data will be peer-reviewed by reviewers at the eLife journal and made available online.

These three initiatives are examples of a large-scale coordinated effort to enhance reproducibility, and many more are on the horizon. So far, we have discussed the problems in the academic environment leading to differing treatment of negative and positive data, the core of reproducibility crisis, and the difficulties that arise as a result. Next, we begin talking about the more serious consequences of data manipulation in the case of drug trials. The data points from clinical trials decide the fate of medications that will affect thousands of lives. Obtaining all relevant data is crucial not only for accurately prescribing medication, but also to avoid pitfalls and avenues already tackled in the past.

■ **Note** Dr. Ben Goldacre gave a TED Talk in which he told a story of the drug lorcainide, released in 1980s. It was meant to be an anti-arrhythmic drug that could prevent abnormal heart rhythms in people who have had heart attacks. A small clinical trial was conducted in fewer than 100 patients, and unfortunately ten of them died. The drug was regarded as a failure and commercial development stopped. The data from this failed clinical trial were never published. Over the next few years, other drug companies had similar ideas for

anti-arrhythmic drugs that were brought to market. It is approximated that 100,000 people died because these new drugs also caused an increased instance of death. In 1993, the researchers who conducted the original 1980 study came forth and wrote an apology mentioning that they attributed the increased death rate to chance in the initial trials. Had the data from this failed trial been published, however, they could have provided early warnings and prevented future deaths. This is just one example of the very serious consequences of publication bias. We tackle a generalized version of this scenario in the next section.

Clinical Trials

We have already described a few complications that arise due to flawed data reporting from clinical trials, and here we begin outlining a potential solution. In this section, we focus on three specific issues and provide a use case for integration blockchain technology in each one.

- *Trial registration*: Registering clinical trials when they begin, providing timely updates, and depositing the relevant results in a public database is crucial for offering clinicians choices in prescribing new medication to patients for whom the standard drugs aren't effective. Even though large-scale clinical trials involving human participants should be registered, more often than not, these trials remain missing in action. The only indications of data coming from the unregistered trial are a publication or perhaps a few papers that contain experiments and results highly tailored toward proving the effectiveness of the candidate drug being proposed. This type of publication bias can mislead clinicians in a dangerous manner; therefore, we need to incentivize investigators to send regular updates from registered clinical trials on progress and any relevant clinical protocols.
- *Comparing drug efficacies*: Today in most clinical settings, multiple drug options are increasingly becoming available to clinicians, but there is often a lack of evidence from head-to-head clinical trials that allows for direct comparison of the efficacy or safety of one drug over another. Computational models allow for parallel processing of large data sets in a type of analysis called mixed treatment comparisons (MTCs). These models use Bayesian statistics to incorporate available data for a drug and generate an exploratory report on the drugs compared. This can become the foundation for automated comparisons, as more data are liberated from unpublished or unavailable information silos.
- *Postprocessing*: In some cases, when a trial is registered, and it does provide some supplemental data that goes along with a publication, the registry acts more like a data dump than an organized data deposition. Recently, we have seen more carefully prepared and published postanalysis summaries, but this is often an exception, not the rule. The key here is that once clinical-trial data are linked up to the blockchain, they become available to be included in an automation workflow. Now postanalysis summaries and data can be generated by an algorithm rather than a person. A universal back end for storage of data can foster the development of front-end clients that read the blockchain and, using the appropriate public–private key pair, download the appended data from an external source and locally do the postprocessing. After that, the summary reports can be appended back to the blockchain.

■ **Note** Soenke Bartling and some collaborators in Heidelberg have been working relentlessly on open-science innovation using blockchain technology. Recently, they founded a think tank called Blockchain for Science to accelerate the adoption of blockchain technology in open science. The interested reader can find more on their web site at `blockchainforscience.com`.

Let's start talking about a viable solution to making clinical trials more transparent using the blockchain. More specifically, making the data from clinical trials available to the blockchain will be done with the implementation of colored coins. The scripting language in Bitcoin allows for attachment of small amounts of metadata to the blockchain. Colored coins are a concept that leverages the blockchain infrastructure to attach static metadata that can represent assets with real-world value. We use colored coins as a metric to introduce scarcity and incentivize upload of additional data, regular updates, and so on. Before we dive more into colored coins, let's look at what makes them special. There are three essential components.

- *Coloring scheme*: The encoding method by the colored coin data is encoded or decoded from the blockchain.
- *Asset metadata*: This represents the actual metadata attached to a colored transaction that gets stored in the blockchain. We go over an example of it later. The new colored coins protocol allows for the attachment of a potentially unlimited amount of metadata to a colored transaction, by using torrents that provide a decentralized way to share and store data.
- *Rule engine*: In the past, the metadata just contained static information added to colored coins. Recently, however, a new rules section has been added that encodes an extra layer of logic supported by our rule engine that unlocks smart contracts' functionality to colored coins. Currently four types of rules are supported, as discussed later.

Here's the generalized syntax for the metadata that can be added to a colored transaction:

```
{
 metadata: {...Static data goes here...},
 rules: {...Rule definitions go here...}
}
```

There are two rules from the rule engine that we will be using in our solution: the expiration rule and the minter rule. The expiration rule is used to loan an asset and it dictates the life span of an asset. After expiration, the asset returns to the last output that had a valid expiration. The minter rule grants a recipient permission to issue more of the same asset. Therefore, the minter receiving colored coins can further issue more colored coins to others on the network. Both rules play an important role for introducing scarcity in this instance of blockchain economics. What role does scarcity play? To understand this, we need to look at two players in this scenario: a holder and a minter. A holder is another rule from colored coins that dictates which address can hold an asset, and we have already described a minter.

Figure 8-2 depicts the interaction between a researcher registering a clinical trial, providing updates, and a minter acknowledging reception of the updates, followed by a request to continue the next phase of the trial. Let's walk through Figure 8-2 step by step. The clinical trial begins with our researcher registering the trial, which initiates a genesis colored transaction from the minter to the holder (our researcher). This transaction comes with an expiration rule attached, and this in a sense is the deadline for one of the several phases related to a given clinical trial. The researcher must send a colored transaction back to the minter, with URLs to metadata containing updates or new data. When the minter receives this transaction, an evaluation is performed on whether the asset was expired, and the result is exported. We return to this result in the next section. After this return transaction, the minter issues more colored coins to the holder for the

next phase, and the cycle repeats. Each phase of the trial results in more data URLs being appended to the metadata sent by the holder as a sign of continuous updates.

■ **Tip** The rules engine of colored coin protocol is also a part of the metadata not stored directly on the blockchain, but instead it is stored in plain JSON format using torrents. Torrents provide a decentralized mechanism to share data, and make it possible for rules to interact with objects outside of the blockchain. We have abstracted the minter here as an oracle, but the actual implementation would involve a hybrid of an automated evaluator and smart contract.

The entire process visualized in Figure 8-2 can be considered analogous to the SSL handshake that forms the foundation of sure server–client interaction and symmetric encryption in our browsers. Perhaps in the future, one can extrapolate; if this type of interaction becomes common, it could become a feature of the colored coin wallet. The evaluation component of the minter address can become either a specialized wallet or a new client, similar to new protocols being included in a browser.

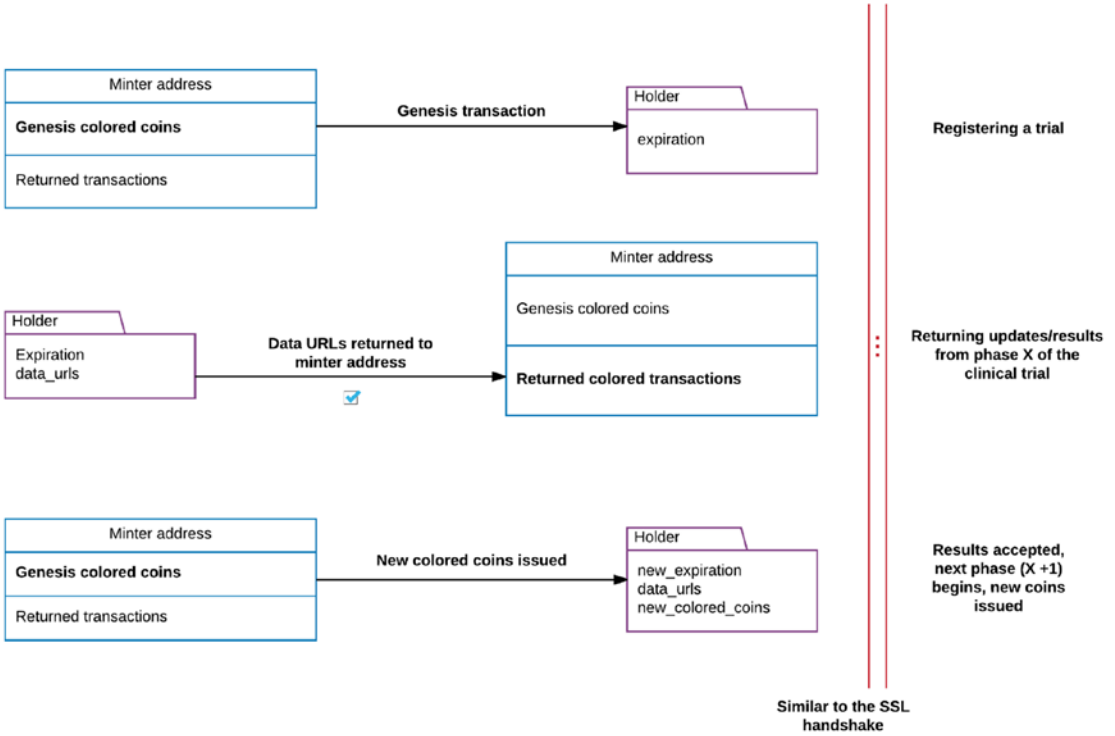


Figure 8-2. *Interactions between a minter and a holder*

This process can create artificial scarcity by imposing the expiration rule. The holder (researcher) has to return a colored transaction with the data URLs corresponding to the update. The minter performs an evaluation on the state of the holder and then acknowledges the receipt of updates. New coins are issued for the next cycle of updates and the whole cycle begins anew. The evaluator result is exported and will be used to build a reputation system that we discuss shortly.

Now that we have discussed the colored coin interactions, let's take a look at the entire clinical trials system in Figure 8-3.

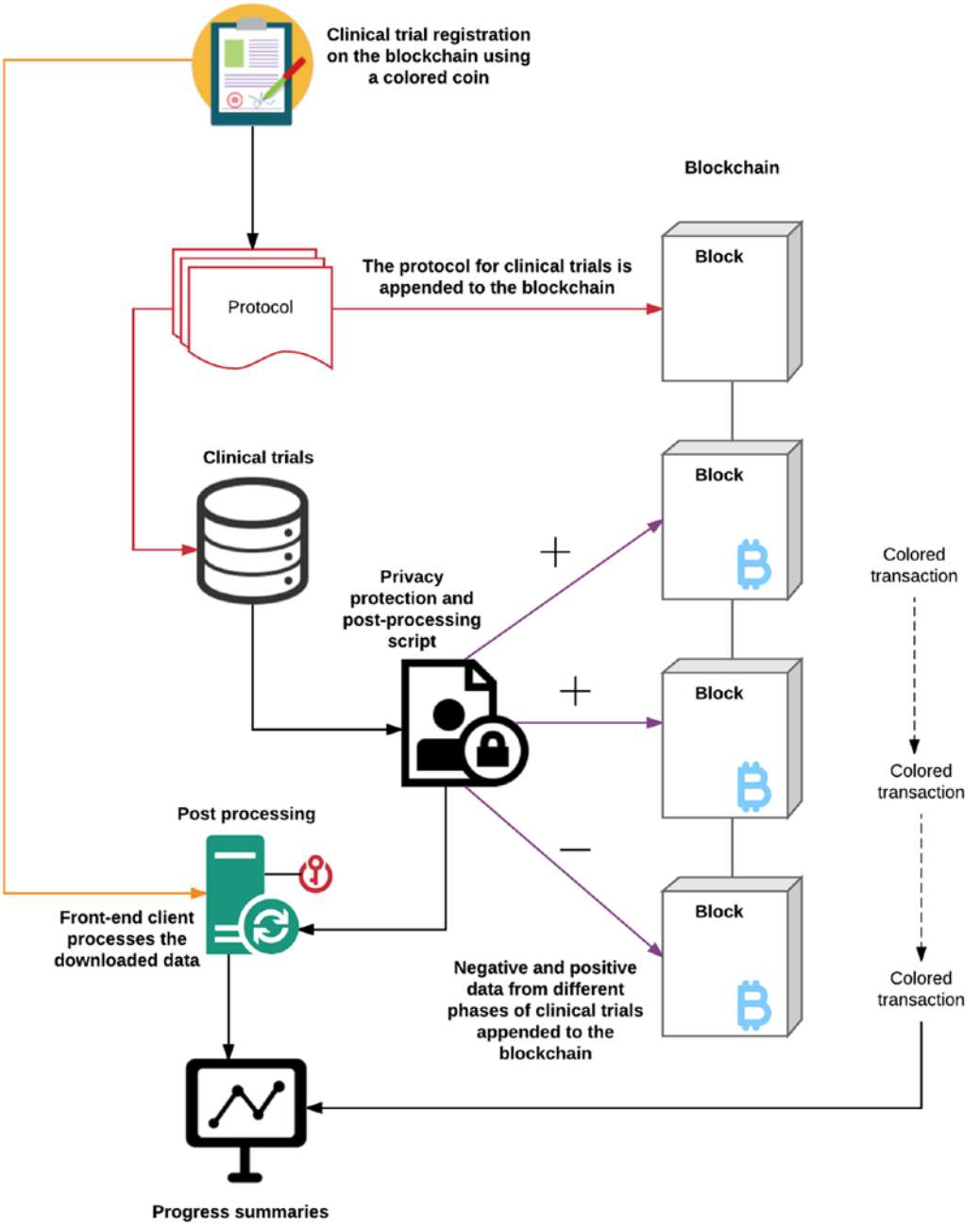


Figure 8-3. *Summary of blockchain integration into clinical trials workflow*

The process begins with the registration and a proposed summary of what the clinical trial will entail, the methods being used, and what data or results can be expected. The summary information along with the approved protocol are appended to the blockchain before the trial begins. This completes the registration process. Colored coins and the rules engine are used to manage updates from researchers. These updates are appended to the blockchain after going through a light privacy protection check. Once the clinical data are on a common back end, the most important benefit is perhaps the shift of focus from back-end clients that hold databases to simply developing front-end clients that can read the blockchain. The management of data will happen automatically within the blockchain; all we need is a mechanism to read the tags or breadcrumbs left over in the metadata to know what to pull from an external location for further processing. An example of this case is the postprocessing unit shown in Figure 8-3. This unit contains the appropriate public–private key pair and permissions to read the blockchain and access the external locations. The same script that appended data updates to the blockchain also contains a segment for postprocessing that tells the postprocessing unit how to integrate data from various third-party locations into one local collection. After that, postanalysis statistical methods are used to determine the quality of the data appended and an automated report can be generated that summarizes the progress of the trial at given intervals. The intervals at which data updates should be required from researchers, along with instructions on how to process those data once made available are coded in a script, made available to the postprocessing unit.

Reputation System

Let's revisit the notion of scarcity. It was crucial for us to build the clinical trials system, however, the introduction of colored coins with the expiration rule also allows us to build another component, the reputation system. The premise of the reputation is simply tracking adherence to the expiration rule. Recall that we built an export feature in the evaluator function and here we can use the export as a counting mechanism to reward researchers (or holder addresses in the colored coin protocol) who have been proactive in sending periodic updates. In a practical application, this export counter would become added to the metadata by the minter after periodic updates have been established. From here, establishing reputation is a straightforward task: A higher export counter corresponds to a better reputation.

It is important to note that for our clinical trials system, reputation simply emerged as a property of the design, but it has some far-reaching implications concerning reproducibility. High reputation indicates the commitment of an institution or a research group toward quality control. Once a reputation mechanism is implemented network-wide on a blockchain, it can be referenced externally: Third-party services can request the reputation score associated with a particular colored wallet. This can be as simple as an API call to return the `rep_score` of a wallet. Why would this be useful? Earlier in our discussion, we mentioned the DDI, and here we want to expand the notion of DDI from just being a data repository to a repository with `rep_score` tags. This can provide another layer of granularity into DDI: a tag of reputation scores (high or low) on data sets linked to clinical trials that are available to the public. Keep in mind that within this design, all the data from clinical trials or updates reside in the DDI repositories; however, the `rep_score` lives in the blockchain metadata through colored transactions. Figure 8-4 describes the sequential evaluation happening with each periodic update and the incremental addition of reputation in `rep_score`.

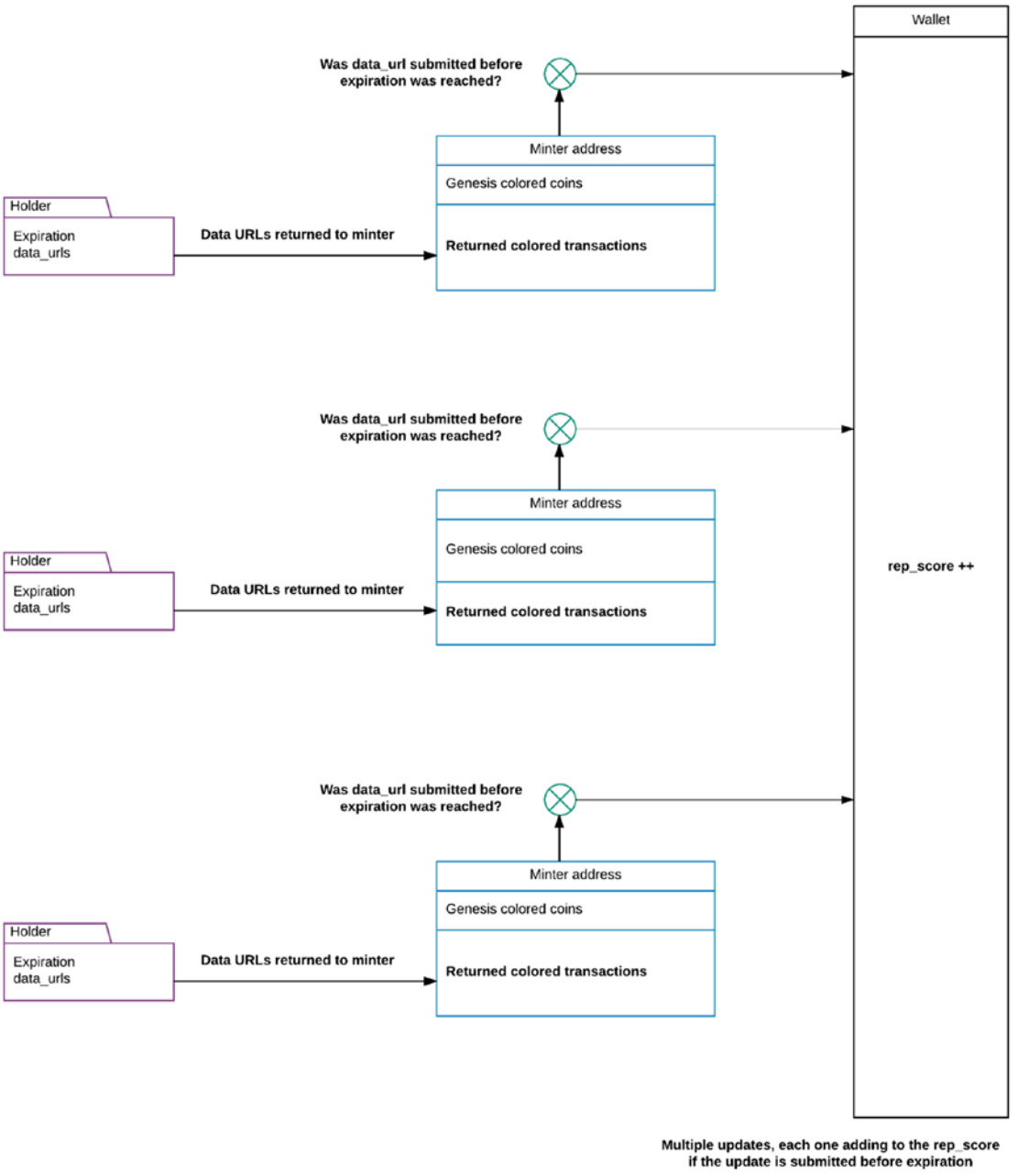


Figure 8-4. *Overview of* `rep_score` *increasing with each periodic update*

The evaluator function checks for an expiration rule and if the colored transactions made by the holder contain the URLs corresponding to updates in the metadata. If those two conditions are met, the `rep_score` is updated for the holder's wallet. This slow increase allows for the reputation to build over time, and the `rep_score` parameter can be referenced in a blockchain agnostic manner from external services. API calls can become the default manner of attaching an up-to-date `rep_score` to databases deposited at DDI.

Now that we have a better understanding of the `rep_score` mechanism, let's look at the complete reputation system. Figure 8-5 provides a comprehensive depiction of a reputation system and its consequences for the members network-wide.

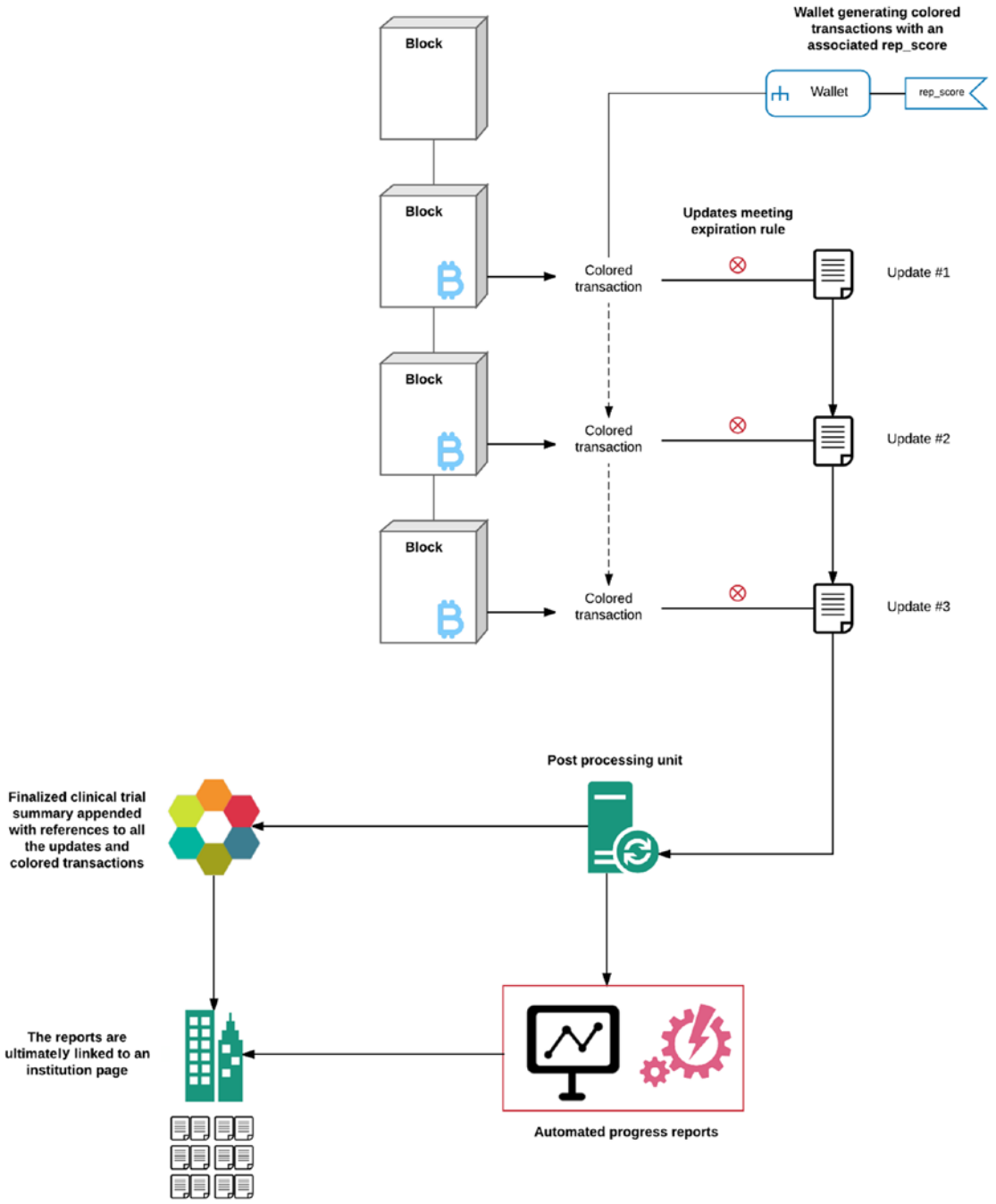


Figure 8-5. *The reputation system*

It begins with the blockchain recording colored transactions happening between the minter and holder roles from colored coin protocol. These colored transactions happen each time the holder provides an update, and the minter evaluates it. After the initial update, the minter sends a `rep_score` as a part of the returned colored transaction to carry on the update process for a clinical trial. The reputation now gets amended to the metadata with each exchange and the `rep_score` gets revised each time. This score can eventually become an attribute of the holder's wallet, and in this manner, it can be referenced externally from the blockchain. The postprocessing unit becomes important after the clinical trial has matured to generate enough updates that can be compiled in one comprehensive overview. The purpose of this unit would be to automatically perform quality control statistical calculations and share the updates following instructions in a script added to the blockchain in the beginning of the trial. In a similar fashion, the postprocessing unit will compile a final summary and attach that to the institution's page, along with all the updates shared from the clinical trial.

■ **Note** It must be noted that postprocessing and data storage are all functions done off the blockchain. No calculations or data manipulations are performed on the blockchain. We want to limit the blockchain-executable code as much as possible to only the necessary primitives. The only tools that interact with transactions or metadata on the blockchain are in place to update the distributed state of a variable or parameter.

Pharmaceutical Drug Tracking

The final use case in this chapter covers tracking pharmaceutical drugs via a supply-chain management framework. Prescription drug monitoring programs are an essential element for controlling drug shopping. The Mayo Clinic defines drug shopping as a patient obtaining controlled substances (most often narcotics) from multiple health care practitioners without the clinician's knowledge of the other prescriptions. This leads to drug abuse or sale on a large scale and the Mayo Clinic attributes drug shopping to the following three factors.

- *Poor timeliness of data*: How often does a pharmacy or provider upload prescription data into their existing centralized system? A blockchain-based back end would make drugs associated with all transactions available instantly to all members of the network.
- *Reliability*: Centralized databases have a single point of failure, as compared to the blockchain, which is decentralized by nature. The data therefore do not rest on a single database, making them more reliable.
- *Complexity of data retrieval*: The current model of data retrieval and compatibility with existing hospital systems is completely broken. Hospitals often are not synced up to the databases being used by pharmacies and updating is an arduous task. Blockchain makes the process universal by providing a common back end to read for external services.

In this system, when a clinician writes a prescription, the provider can check the blockchain to find the patient record for any currently active prescriptions. This can help the clinician determine whether the patient is asking for prescriptions from multiple sources or whether another family member is getting the same prescription. An active prescription from a different provider would automatically invalidate the request for a new one, and this could be encoded in the network as a double spending request. Otherwise, the transaction will go through and the pharmacy will receive a request to provide medication to the patient. The requesting provider and patient can sign the transaction, allowing for better tracking all the way through the care of a particular patient. Although this system seems simple, it fulfills most of the requirements for

a Drug Supply Chain Security Act (DSCSA) compatible system that can be implemented across providers. There are some efforts underway to control the opioid overprescription epidemic by major blockchain players such as IBM and Deloitte blockchain lab. Recently, some new startups have sprung up focused solely on pharmaceutical drug tracking using the blockchain.

Even though supply-chain management fits the blockchain infrastructure most closely and out of the box, this field is very new and the technology is immature. The future of drug tracking and open science looks much more promising with new technologies enabling layers of communication that were never possible before. Blockchain, if nothing more, has been an immense catalyst of new innovation in open scientific inquiry and discourse.

Prediction Markets and Augar

In this chapter, we talked about reputation being a key parameter in holding researchers accountable for providing additional scientific data that can enhance the reproducibility of published studies. A crucial challenge to achieving this goal is the alignment of incentives for researchers to gain benefits from making their data available. Let's look at some current efforts in academia that would make integration of a blockchain in existing infrastructure more feasible.

In 2015, a large-scale crowd-funded replication effort was underway to verify key findings from 100 psychology studies. Before this effort concluded, some of the researchers filled out a survey for 44 studies indicating the likelihood they thought that the study's main finding would be successfully replicated. The same researchers also played a betting game: They were given $100 USD to buy and trade contracts in the ongoing replication efforts. Each contract would pay $1 if the key finding were to be successfully replicated. Data from the survey and the betting games was collected and published in Proceedings of the National Academy of Sciences (PNAS). It turns out that the survey predictions were about as accurate as a coin flip. However, if the researchers engaged in a betting market on the same question, the data gave a good prediction (with approximately 71% accuracy) of which studies would hold up if replicated. The market approach might have been more accurate than the survey for two reasons: First, there were monetary incentives to make the best bet, and second because the market enables people to learn trends from other players and to adjust their bets. The authors of the PNAS study suggest that prediction markets can be used to decide which studies should be prioritized in replication efforts due to their uncertainty. They warn that prediction markets will not reliably predict which individual findings are accurate, but they can be used to estimate the credibility of current findings and make an ordered queue of studies to be replicated.

Setting up a prediction market would require a platform, and that's where Augar comes in. Augar is a decentralized prediction market platform that runs on the Ethereum blockchain. The underlying exchange of value on Augar is REP tokens or reputation. REP is used by referees (also known as reporters) to report event outcomes on the platform, and Ether is used by other users to place bets on markets.

On Augar, the prediction market works in two phases: Forecast and reporting. Initially, a user supplies some funds to creates a prediction market to address an event outcome or question, and the forecasting period begins. Other users place their bets during this period and the event matures. After the event has happened, the reporting period begins. The reporters act as oracles for the network and submit their answers or outcome. After a sufficient number of answers have been received, a consensus on the event is generated. The reporters who submitted the correct outcome are rewarded with reputation for their service, and the users who made a bet on the correct outcome are paid out in Ether.

The betting games played by researchers in the PNAS study can be repeated on a platform like Augar at a larger scale. More work remains to be done before Augar can be used for prediction in replication studies, but the platform has all the required features to serve large-scale reproducibility efforts.

Summary

This chapter started with a broad description of the reproducibility problem and its serious economic consequences in evidence-based sciences. Then, we moved into discussing the current solutions and their shortcomings in the science community. After that, we described the idea of building reputation systems using blockchain and covered three use cases: clinical trials, reputation networks, and finally pharmaceutical tracking of drugs from manufacturing. All the use cases highlighted the strengths of blockchain in tracking and accountability over traditional methods.

CHAPTER 9

Blockchain in Health Care

The health care sector is a $3 trillion industry, and about $1 trillion of it goes to waste annually. Care coordination is becoming more complex as chronic conditions in the aging population continue to rise. In many instances, the technology available to health care providers is not adequate enough to capture all aspects of the care being provided. The result is a rugged transfer of information between parties that ultimately reduces the quality of care being provided to patients. This is largely due to providers having legacy systems, lack of integration with non-vendor-specific technologies, paper-based medical records, and a lack of horizontal transfer between allied health care professions. Hospitals are investing a considerable amount of resources into processing medical claims and administrative records when most of this overhead can be eliminated by using a sophisticated technology infrastructure. In this chapter, we discuss the payer–provider–patient model in the context of the incentives and services each provides and how this model is likely to change in the near future. The core of this chapter is constructing a conceptual workflow showing how blockchain can integrate into tracking a patient from the first visit to a primary care physician to the final diagnosis and a treatment plan. Then, we will introduce the topic of hot-switching, and briefly describe two new implementations of hot-switching enabled on blockchain: Lightning network and pluggable consensus. Finally, we will end with talking about waste management in healthcare and the efforts by Capital One + Ark Invest to reduce the economic inefficiencies applying blockchain to business processes.

Payer–Providers–Patient Model

The payer–provider–patient model is a standard model of interaction among the three major players in health care. Figure 9-1 shows these interactions visually, and in this section, we elaborate on the incentives and benefits for these three players. For the sake of simplicity, let's assume that the payers are largely insurance companies, the providers are hospital systems or private clinics, and patients are a randomized sample served by a given insurance company.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_9

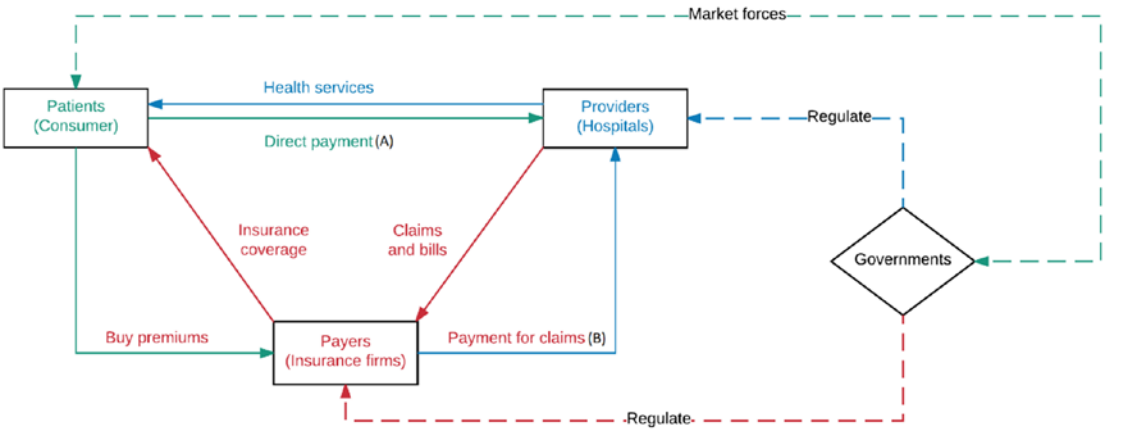


Figure 9-1. *A simplified overview of the payers–providers–patients model*

Now we can start to consider the different scenarios involving the model shown in Figure 9-1.

- The first scenario is the simplest one, including the patient directly paying for procedures and visits to the hospital. This straightforward system involving just two players is still the norm for numerous health care systems in different countries. In the United States, the cost of providing health care has increased significantly, so a new player has emerged within the system.
- The second scenario is the current implementation across the United States. The patients are provided health coverage after buying a premium from an insurance company. This company then gets more negotiating power to set viable prices for procedures and then pay for them on behalf of the patients. The providers now send the claims for payment to the insurance companies, instead of the patients directly. In this manner, the insurance companies have evolved to become a centralized portal for patients to remove friction out of the system and receive personalized health care.
- The third scenario is more about the future of health care with the integration of new providers that are specialized entities for administering care through telehealth, off-site monitoring, and remote physicians.

The simplest case is the patient directly paying for services to the hospital. More complicated medical procedures and lab tests, however, require some coverage that the patient purchases from an insurance company. Now the insurance company pays on behalf of the patient for any medical bills. The provider also sends claims for payments directly to the insurance company, instead of the patient. This model is foundational to understanding the immense complexity present in the health care system today, which involves hundreds of other players. For instance, the domain in which insurance companies and hospitals can operate with respect to patients is largely regulated by governmental agencies. These agencies are in turn influenced by economic forces and consumer demand.

■ **Tip** Chris Kay, CIO of Humana, delivered the keynote address at Distributed:Health in 2016 discussing Humana's efforts toward adopting blockchain to reduce processing costs in health care and providing personalized care more efficiently to their members. He spoke about a potential path of evolution for the second scenario mentioned earlier wherein the blockchain can power the flow of payments between providers and payers seamlessly and drive accountability toward outcomes, not just procedures. Shifting the locus of control to the patient allows them to receive better care and satisfaction, without any increase in friction or overhead costs using the blockchain.

The idea behind claims processing on the blockchain is to use features such as decentralized consensus, trustless transactions, and network verification to design workflows that reduce overhead costs and time. We wanted to address the following question: Can we design the concept of a patient workflow based on the blockchain that documents the transition of a patient from her primary care physician, to lab-work, and finally to a specialist? Let's take a look at a patient visit workflow involving the use of blockchain.

Workflow

In this section, we want to design a rudimentary Electronic Health Record (EHR) system based on access permissions being passed from one party to another using public/private keypairs. Our workflow begins with Jane visiting her primary care physician (PCP) for a checkup following gastric discomfort after eating certain foods. Her physician suspects a viral infection and orders further lab work to be performed. She is further referred to a specialist, who gives the final diagnosis, and works with her to develop a treatment plan. Every step of this workflow is recorded on the blockchain as Jane's medical record, with appropriate permissions regarding access rights and ownership. Additionally, Jane's records would also contain a history of every check-out and check-in, including edits and additions that were made to the medical record. Closely tracking how the ownership changes helps us transition between the different points of care provided to Jane, and she has instant access to her medical records as a clearer picture of her health begins to emerge.

The main objective of this workflow is to demonstrate the concept of how blockchain handles permissions and transfer of ownership between multiple unrelated parties. The use of user signatures using cryptographic keys is leveraged in creating this rudimentary medical record system. We can understand the attributes and permissions within our workflow better using the terminology of version control systems. Just as Git or SVN have commit messages and code commits, every check-in of new data to the medical record will contain a commit message about what changes were made. The blockchain becomes a remote repository of sorts (like GitHub) to which Git can push new changes. There are also locking mechanisms to protect the data: Once a document from the medical record is checked out to a user (a doctor or a nurse), it will be locked and no other user can check in new changes to the same document until the original user checks in his or her edits, unlocking the document. Figures 9-2 through 9-4 guide us through this workflow.

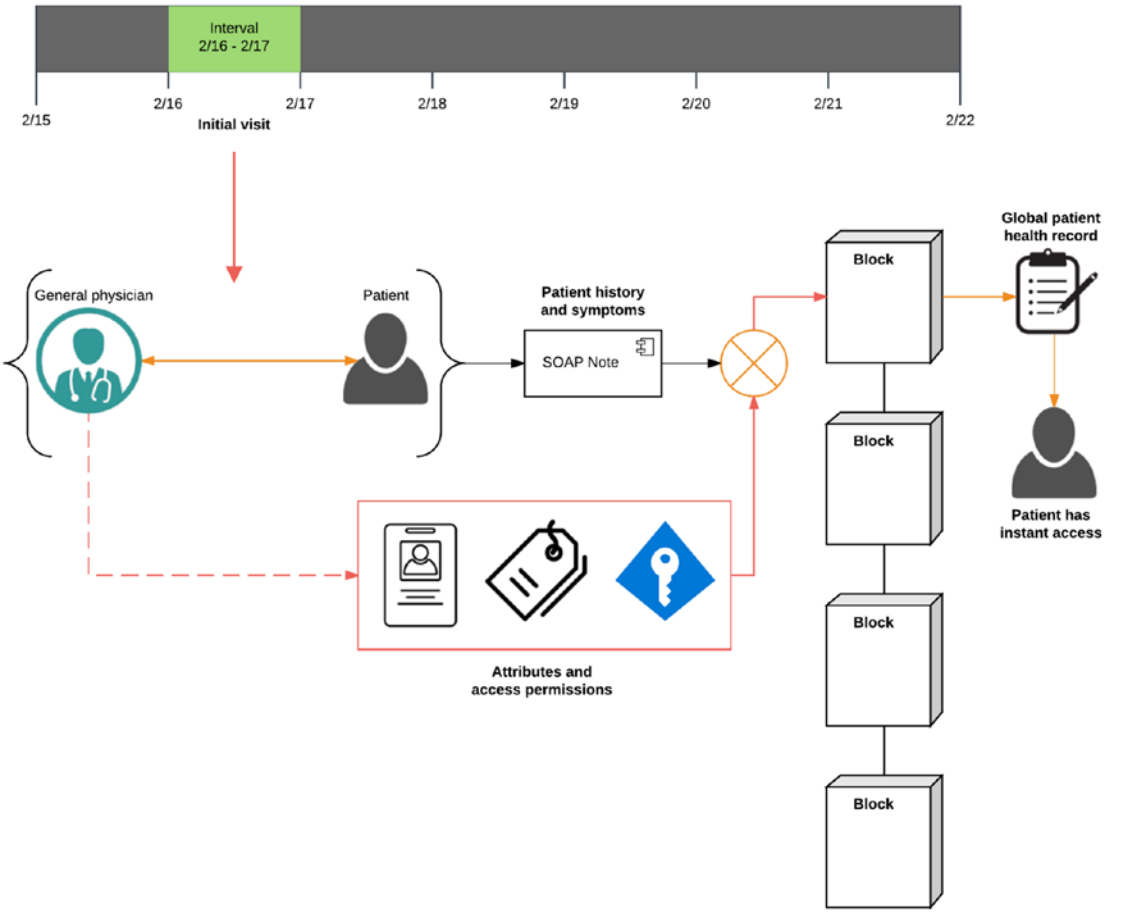


Figure 9-2. *Initial patient visit*

The workflow begins with Jane visiting her general physician for a regular checkup. The physician takes a history and records any symptoms that stand out from a review of her body systems. He adds this information to a very standard method of documenting patient history called a SOAP note. The note will be added to the blockchain with a unique ID called the hash. Along with the hash, the physician will add permission and user group roles for accessing the medical record. Initially, only the physician and the patient would have access, but this can be expanded easily. Finally, the note is signed using the physician's key to denote the initial commit of Jane's medical record to the blockchain. Once Jane's records are pushed to the blockchain, she also has instant access to them.

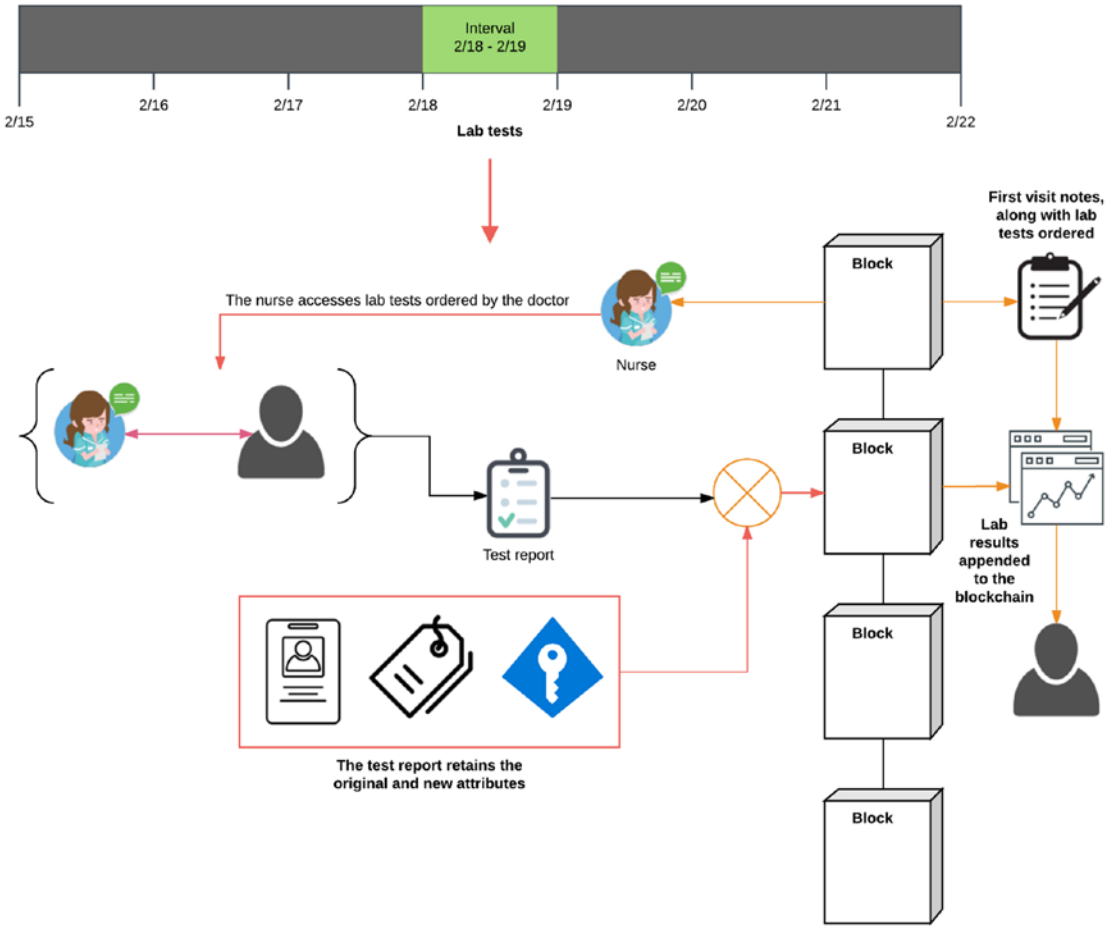


Figure 9-3. *Jane's physician ordered some lab tests to be performed so two days later, she visits a lab*

The tests have been entered into her medical record along with the permissions to let a nurse at the lab access the details. The physician can easily set the access permissions and add new upload privileges for the nurse on the medical record. This workflow begins before Jane's arrival, when the nurse retrieves Jane's record and begins preparing for the tests. When Jane arrives, they have a consult and the nurse informs her of the tests she will be going through. After the tests, a report is generated and appended to Jane's records on the blockchain. We can see this on the right side where Jane's general record now has a new add-on for the lab tests recently performed. All these results are instantly available to Jane as they are uploaded.

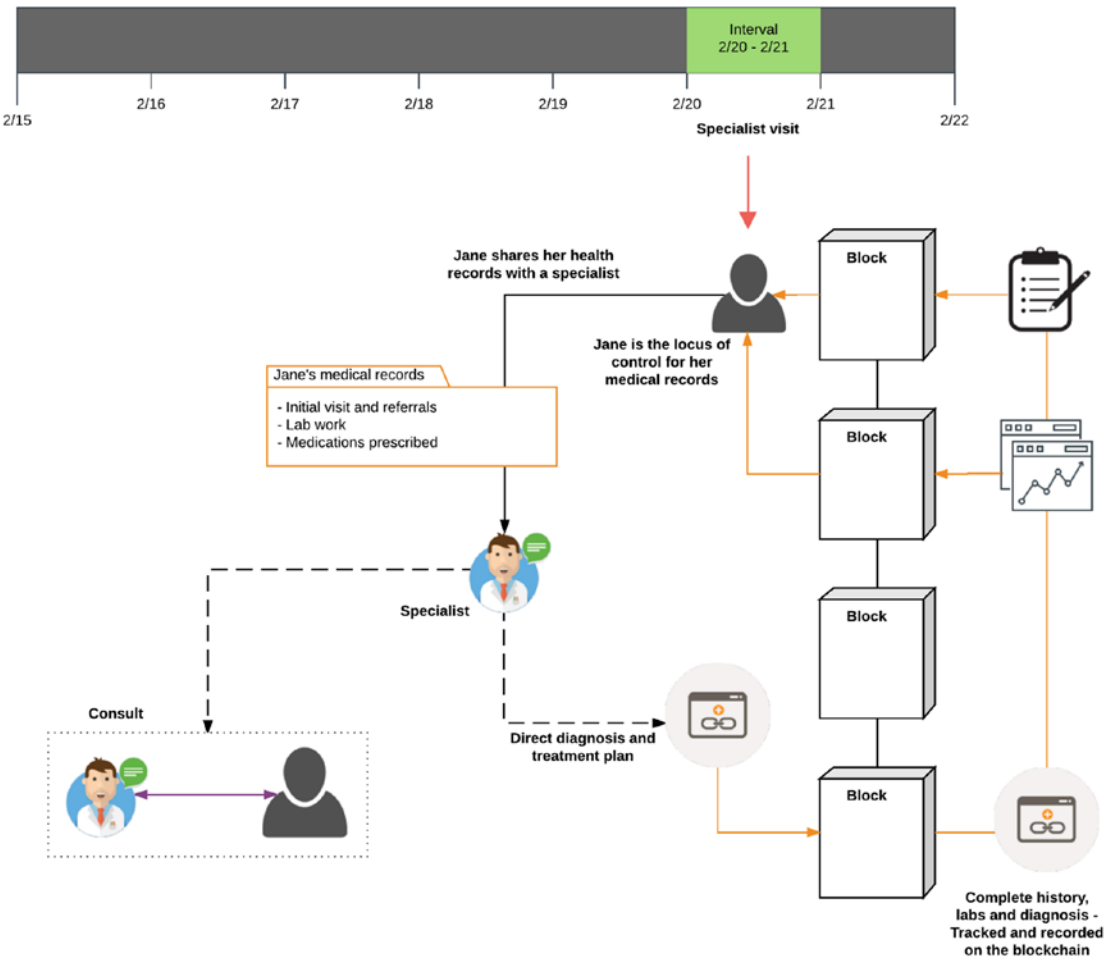


Figure 9-4. *Specialist integration*

In this step of the workflow, Jane decides to visit a specialist to better understand the lab results. Two days later, she shares her medical records with a specialist. At this point, the specialist has two choices: The specialist can examine her records and provide a diagnosis along with a treatment plan for her without even needing a visit. Alternatively, Jane can go in for a consult and work with the specialist to adopt the treatment plan. The first option might help reduce the cost of visits and services received for Jane, as ultimately the treatment plan and diagnosis will both be made available to her on the blockchain. This completes the workflow from her initial visit to the point that she received a diagnosis. Every step of this workflow from the

original notes to the final treatment plan were recorded on the blockchain. However, in the case of a patient with a chronic disease, the records from this point would be transferred to a chronic care facility that can check up on Jane regularly and help her maintain good health.

Hot Switching

Hot switching is a design principle enabled by the blockchain to create systems where components can be swapped out while the system is still running with minimal latency in operations. The purpose of hotswitching is to re-route the flow of information, and this concept is only made possible due to the decentralized nature of the blockchain, that is to say, there is no single point of failure. Why do we need a new concept such as hotswitching at all? Traditional schedulers perform this job fairly well. An operational blockchain-based medical records system will require both live and off-chain components that are synced to the blockchain. IT infrastructure in hospitals should be very stable with minimal downtime, but hotswitching can enable isolated system upgrades without disruptions, and gradually transition the legacy systems that are still active to be compatible with the blockchain.

There are two early implementations of hotswitching available: The Lightining network (`https://lightning.network/`), and pluggable consensus. So, what is this Lightning network? The alpha-version of Lightining offers a mechanism to send transactions over new micropayment channels (sub-channels). This transfer of value occurs off-blockchain thereby making transactions instantenous. These channels ultimately sync upstream with the blockchain and preserve data-integrity of the overall network. In a similar fashion, off-chain channels can replicate and temporarily store the intermittent requests for access and release medical records based on pre-approved access controls. After some defined period, the lightning channel can sync the intermediate changes with blockchain and become inactive. The second application of hotswitching is pluggable consensus, which refers to the idea that you can swap out your consensus algorithm based on the types of transactions being handled on the blockchain. For instance, private transactions carried out on a public/private partitioned blockchain may reuqire a different consensus mechanism than general purpose transactions. Therefore, multiple consensus algorithms are allowed on a blockchain. We will discuss the implementation of pluggable consensus in more detail in a later chapter.

Waste Management: Capital One, Ark Invest, and Gem

A webinar hosted by Gem, Capital One and Ark Invest outlined the issues of economic costs and overheads plaguing the healthcare system. This economic waste was highlighted in the context of claims processing from the perspective of payers (insurance companies). However, the lessons learned here can be broadly applied to other industries processing claims such as auto-insurance. As an example, Gem announced a partnership with Toyota to port applications it has been developing for the healthcare insurance industry into car insurance. The aspiration is to automate most of the insurance claim process using a blockchain. Here, we will summarize the key findings outlined by the panelists. In the following discussion, each problem raised by the panelists is represented by a [P] and a proposed solution is represented with a [S]:

- **[P]** For every dollar collected by healthcare providers, 15 cents go towards claims processing, facilitating payments between parties and manual labor costs. Imagine the 15 cents adding up to a $3 trillion industry..
- **[P]** Providers respond by increasing reimbursement rates and payers (insurance companies) respond by increasing premiums.
- **[P]** Hospitals are acting as banks by giving loans to patients for out-of-pocket payments and lacking the infrastructure to support and absorb inefficent payment frequencies.

- **[P]** Hospitals only have a collection rate of 5% on patient out-of-pocket payments that were provided as loans.
- **[P]** These gross inefficiencies are leading to hospitals charging up to 45% more premiums on top of the base rate for services.
- **[S]** On the bright side, reducing the time it takes to process claims from providers can offer $23 billion in savings.
- **[S]** Decreasing the volatility of collections in the billing cycle from out-of-pocket charges can offer $7 billion in savings.
- **[S]** Using the blockchain to track transactions with history can significantly decrease rate of fraud, saving insurance companies approximately $300 billion.
- **[P]** Claims clearing houses have to spend days, and sometimes weeks to process claims, adding significant overheads.
- **[P]** Providers are using multiple third party software tools to manage claims. Instead of the technology being assistive, clinics are experiencing more fragmentation and using manual labor to integrate different software.
- **[S]** The blockchain can track the entire continuum of care and the billing cycle to reduce friction between the involved parties.
- **[S]** On a decentralized ledger, transactions and claims can be tracked very efficiently leading to claims resolution almost instantenously, if not in a few minutes.
- **[S]** Public ledgers such as the Bitcoin blockchain can't be used, HIPPA laws limit how patient information can be transferred across digital channels. The eventual solution would be to use a permissioned ledger with verified nodes. This will lead to restricted access by permissioned users to medical records being tracked by public and private keys usage.
- **[S]** Smart contracts responsible for coordinating permissions and access roles on the blockchain between various parties requesting medical records. Not every party can see the entire record, permissioned access allows for only relevant exposure.
- **[P]** Providers are not IT professionals, not enough time to provide training for new systems therefore interoperatiblity remains a major issue.
- **[P]** Lack of interoperability spans to:
    - Patient registration in a medical system
    - Authorization of procedures
    - Medical records
    - Co-pay collection
    - Claims submission
    - Patient billing
    - Accounting of claims and bills

- **[P]** Functionality and partitioning - Should claims and medical records be on the same blockchain? The panelists talked about a scenario at Aetna, receiving about a million faxes a year from providers regarding claims information. This adds a massive administrative overhead to manage the paperwork. Ideally, both claims and records go hand in hand. A procedure done on a patient uploaded to the blockchain should trigger a claim to be processed. Ultimately, with proper partitioning, having both on the blockchian would be more efficient.
- **[S]** The concept of pegged-sidechains may become more relevant for coordinated care involving multiple parties, as shown below in Figure 9-5. A domain-specific sidechain could facilitate the transfer of access better, and then sync with the main blockchain. A very basic schematic of a sidechain is presented in Figure 9-6.

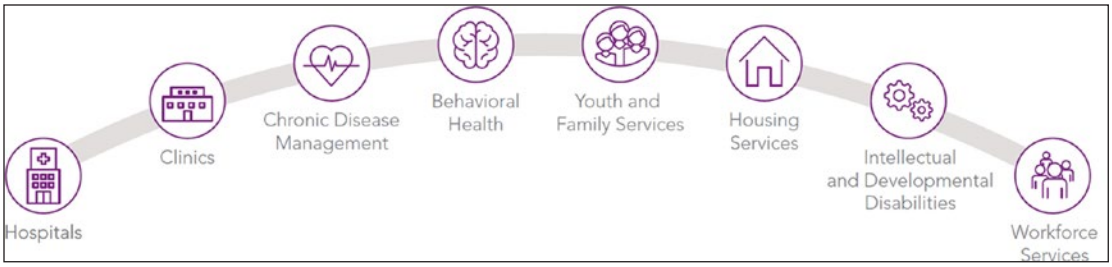


Figure 9-5. *Continuum of Care. This image shows the comprehensive array of health services spanning all levels of care providers and community services. This picture was taken from a presentation done by Eccovia on accountable care organizations, with permission. Done by Eccovia on accountable care organizations, with permission.*

Blockchain can streamline the flow of information for medical records as it switches hands across multiple parties, and in this section, we summarized some of the key findings of economic waste discovered by Ark Invest and Capital One. Even though sidechains are still very early in development, a simple graphic is presented below (in Figure 9-6):

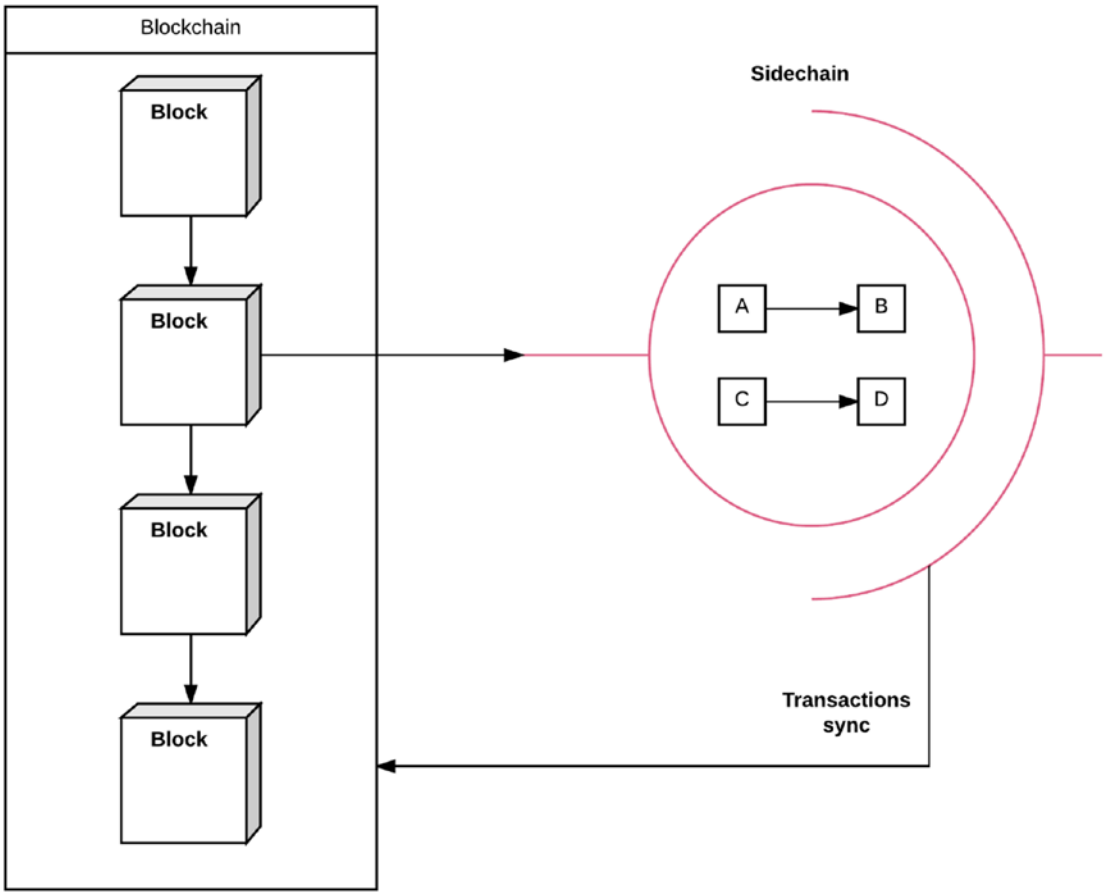


Figure 9-6. *A simple sidechain performing off-chain transactions. This process can be useful for rapidly granting permissioned access and privileges, eventually these transactions sync with the upstream.*

Verifiable Data Audit

Google's DeepMind is working on an interesting blockchain-like service to function as an audit log for hosptials using DeepMind's health services to assist clinical diagnosis. The hospitals are transferring sensitive patient data over to DeepMind's machine learning and AI services to make clinical predictions, and this project is an effort to ensure use of data is compliant with patient consent. We reached out to DeepMind, inquiring about their efforts, and with permission, we're reproducing an overview of Verifiable Data Audit from Deep Mind's blog post:

> *Data can be a powerful force for social progress, helping our most important institutions to improve how they serve their communities. As cities, hospitals, and transport systems find new ways to understand what people need from them, they're unearthing opportunities to change how they work today and identifying exciting ideas for the future.*

Data can only benefit society if it has society's trust and confidence, and here we all face a challenge. Now that you can use data for so many more purposes, people aren't just asking about who's holding information and whether it's being kept securely – they also want greater assurances about precisely what is being done with it.

In that context, auditability becomes an increasingly important virtue. Any well-built digital tool will already log how it uses data, and be able to show and justify those logs if challenged. But the more powerful and secure we can make that audit process, the easier it becomes to establish real confidence about how data is being used in practice.

Imagine a service that could give mathematical assurance about what is happening with each individual piece of personal data, without possibility of falsification or omission. Imagine the ability for the inner workings of that system to be checked in real-time, to ensure that data is only being used as it should be. Imagine that the infrastructure powering this was freely available as open source, so any organisation in the world could implement their own version if they wanted to.

The working title for this project is "Verifiable Data Audit", and we're really excited to share more details about what we're planning to build!

Verifiable Data Audit for DeepMind Health

Over the course of this year we'll be starting to build out Verifiable Data Audit for DeepMind Health, our effort to provide the health service with technology that can help clinicians predict, diagnose and prevent serious illnesses – a key part of DeepMind's mission to deploy technology for social benefit.

Given the sensitivity of health data, we've always believed that we should aim to be as innovative with governance as we are with the technology itself. We've already invited additional oversight of DeepMind Health by appointing a panel of unpaid Independent Reviewers who are charged with scrutinising our healthcare work, commissioning audits, and publishing an annual report with their findings.

We see Verifiable Data Audit as a powerful complement to this scrutiny, giving our partner hospitals an additional real-time and fully proven mechanism to check how we're processing data. We think this approach will be particularly useful in health, given the sensitivity of personal medical data and the need for each interaction with data to be appropriately authorised and consistent with rules around patient consent. For example, an organisation holding health data can't simply decide to start carrying out research on patient records being used to provide care, or repurpose a research dataset for some other unapproved use. In other words: it's not just where the data is stored, it's what's being done with it that counts. We want to make that verifiable and auditable, in real-time, for the first time.

So, how will it work? We serve our hospital partners as a data processor, meaning that our role is to provide secure data services under their instructions, with the hospital remaining in full control throughout. Right now, any time our systems receive or touch that data, we create a log of that interaction that can be audited later if needed.

With Verifiable Data Audit, we'll build on this further. Each time there's any interaction with data, we'll begin to add an entry to a special digital ledger. That entry will record the fact that a particular piece of data has been used, and also the reason why - for example, that blood test data was checked against the NHS national algorithm to detect possible acute kidney injury.

The ledger and the entries within it will share some of the properties of blockchain, which is the idea behind Bitcoin and other projects. Like blockchain, the ledger will be append-only, so once a record of data use is added, it can't later be erased. And like blockchain, the ledger will make it possible for third parties to verify that nobody has tampered with any of the entries.

But it'll also differ from blockchain in a few important ways. Blockchain is decentralised, and so the verification of any ledger is decided by consensus amongst a wide set of participants. To prevent abuse, most blockchains require participants to repeatedly carry out complex calculations, with huge associated costs (according to some estimates, the total energy usage of blockchain participants could be as much as the power consumption of Cyprus). This isn't necessary when it comes to the health service, because we already have trusted institutions like hospitals or national bodies who can be relied on to verify the integrity of ledgers, avoiding some of the wastefulness of blockchain.

We can also make this more efficient by replacing the chain part of blockchain, and using a tree-like structure instead (if you'd like to understand more about Merkle trees, a good place to start would be this blog from the UK's Government Digital Service). The overall effect is much the same. Every time we add an entry to the ledger, we'll generate a value known as a "cryptographic hash". This hash process is special because it summarises not only the latest entry, but all of the previous values in the ledger too. This makes it effectively impossible for someone to go back and quietly alter one of the entries, since that will not only change the hash value of that entry but also that of the whole tree.

In simple terms, you can think of it as a bit like the last move of a game of Jenga. You might try to gently take or move one of the pieces - but due to the overall structure, that's going to end up making a big noise!

So, now we have an improved version of the humble audit log: a fully trustworthy, efficient ledger that we know captures all interactions with data, and which can be validated by a reputable third party in the healthcare community. What do we do with that?

The short answer is: massively improve the way in which these records can be audited. We'll build a dedicated online interface that authorised staff at our partner hospitals can use to examine the audit trail of DeepMind Health's data use in real-time. It will allow continuous verification that our systems are working as they should, and enable our partners to easily query the ledger to check for particular types of data use. We'd also like to enable our partners to run automated queries, effectively setting alarms that would be triggered if anything unusual took place. And, in time, we could even give our partners the option of allowing others to check our data processing, such as individual patients or patient groups.

The Technical Challenges Ahead

Building this is going to be a major undertaking, but given the importance of the issue we think it's worth it. Right now, three big technical challenges stand out.

No blind spots. For this to be provably trustworthy, it can't be possible for data use to take place without being logged in the ledger - otherwise, the concept falls apart. As well as designing the logs to record the time, nature and purpose of any interaction with data, we'd also like to be able to prove that there's no other software secretly interacting with data in the background. As well as logging every single data interaction in our ledger, we will also need to use formal methods as well as code and data centre audits by experts, to prove that every data access by every piece of software in the data centre is captured by these logs. We're also interested in efforts to guarantee the trustworthiness of the hardware on which these systems run - an active topic of computer science research!

Different uses for different groups. The core implementation will be an interface to allow our partner hospitals to provably check in real-time that we're only using patient data for approved purposes. If these partners wanted to extend that ability to others, like patients or patient groups, there would be complex design questions to resolve.

A long list of log entries may not be useful to many patients, and some may prefer to read a consolidated view or rely on a trusted intermediary instead. Equally, a patient group may not have the authority to see identified data, which would mean allowing our partners to provide some form of system-wide information - for example, whether machine learning algorithms have been run on particular datasets - without unintentionally revealing patient data.

For technical details on how we could provide verified access to subsets or summaries of the data, see the open source Trillian project, which we will be using, and this paper explaining how it works.

Decentralised data and logs, without gaps. There's no single patient identified information database in the UK, and so the process of care involves data travelling back and forth between healthcare providers, IT systems, and even patient-controlled services like wearable devices. There's a lot of work going into making these systems interoperable (our mobile product, Streams, is built to interoperable standards) so they can work safely together. It would be helpful for these standards to include auditability as well, to avoid gaps where data becomes unauditable as it passes from one system to another.

This doesn't mean that a data processor like DeepMind should see data or audit logs from other systems. Logs should remain decentralised, just like the data itself. Audit interoperability would simply provide additional reassurance that this data can't be tampered with as it travels between systems.

This is a significant technical challenge, but we think it should be possible. Specifically, there's an emerging open standard for interoperability in healthcare called FHIR, which could be extended to include auditability in useful ways.

Building in the Open

> *We're hoping to be able to implement the first pieces of this later this year, and are planning to blog about our progress and the challenges we encounter as we go. We recognise this is really hard, and the toughest challenges are by no means the technical ones. We hope that by sharing our process and documenting our pitfalls openly, we'll be able to partner with and get feedback from as many people as possible, and increase the chances of this kind of infrastructure being used more widely one day, within healthcare and maybe even beyond.*

Summary

This chapter focused on the emerging role of blockchain in payments processing and how that can be applied to health care. We began the discussion with a description of the centralized payer–provider–patient model and how that model will change in the near future. Then, we discussed how to build a simplistic electronic health record system using the blockchain. After that, we presented how a patient workflow can be recorded on the blockchain. Finally, we talked about the economic wastes in health care discussed by Ark Invest and Gem and how the blockchain can alleviate some of those strains in the near future.

References

The main references used to prepare this chapter were Gem's blog posts on blockchain-based EHRs.

CHAPTER 10

The Hyperledger Project

The Hyperledger Project is a Linux Foundation initiative to develop an open source ecosystem of blockchain development. The Linux Foundation aims to create an environment in which communities of software developers and companies meet and coordinate to build blockchain frameworks. Hyperledger itself is not another cryptocurrency, but rather an open hub for enterprise-grade blockchain projects to incubate and mature through all stages of development and commercialization. In this chapter, we talk about the current state of the Hyperledger Project, with a focus on the currently incubating projects, a summary of the project scope being implemented, and a review of the comprehensive set of technologies involved in creating an open source enterprise-grade blockchain.

Current Status

Under the Hyperledger Project, eight projects are currently incubating: There are five frameworks in the pipeline, and three development tools that go with those frameworks. In this section, we will first provide a brief summary of all eight projects, and then focus on two specific blockchain platforms - Fabric and Sawtooth. Let's begin with an overview:

- *Sawtooth:* This is an Intel effort to create an open source blockchain. Hyperledger Sawtooth is a modular platform for building, deploying, and running distributed ledgers. It includes a new consensus algorithm called Proof of Elapsed Time (PoET), which targets large distributed validator populations with minimal resource consumption. It allows both permissioned and permissionless ledgers to be deployed on a large scale.
- *Iroha:* This is a coherent set of libraries and components that will allow adoption of distributed ledger technologies into existing infrastructure. There is a heavy focus on mobile libraries and mobile applications. Data storage and synchronization done off-device and a default reputation system are implemented network-wide to ensure validated nodes.
- *Fabric:* This is a blockchain development framework for developing enterprise applications with a modular architecture. It is intended to be a component-based system with an emphasis on plug-and-play features such as pluggable consensus, and unique membership services for different user roles.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_10

- *Burrow:* This permissioned blockchain node executes smart contracts in a similar manner to the EVM. Burrow is built to be executed in a multichain environment with application-specific smart contracts. A Burrow node can provide contract execution services to multiple connected blockchains compatible with each other but running a different domain. A Burrow node has three main components: the consensus engine, the permissioned EVM, and a remote call gateway to execute contracts.
- *Indy:* This is a Hyperledger software development kit (SDK) that allows self-sovereign identity to be built into distributed ledgers. This SDK provides wrappers for many languages to add new features and build better decentralized identity managers.
- *Composer:* This is a Hyperledger front-end interface to build and deploy simple blockchain networks for specific use cases. Hyperledger Composer allows us to write simple smart contracts and deploy them to an internal blockchain network. In a large development setting, only a few core users would update the code for a ledger such as Fabric. Most users would be working within Composer to access the blockchain and carry out the daily activities of accessing and updating the blockchain.
- *Explorer:* Similar to any traditional blockchain explorer, Hyperledger Explorer allows a user to query blocks, search through transactions and associated data, network health and information, types of smart contracts being executed, and transaction families stored in the ledger. This project will be compatible with any blockchains deployed with Hyperledger, so new projects do not have to design a new explorer module.
- *Cello:* This is a management tool for deployments of blockchain-as-a-service instances. Cello simplifies the process of creating blockchain instances and reduces the effort required for creating, managing, and terminating blockchains. It provides a containerized blockchain service that can be deployed to existing infrastructures on the cloud, bare metal, virtual machines, or special container platforms.

Governance

These eight projects are all under the umbrella of Hyperledger, and hosted by the Linux Foundation. How does the Linux Foundation support the Hyperledger Project as a parent organzation? It helps by providing governance, both technical and legal governance. The Linux Foundation has a deep understanding of how to build open source communities, and as the incubating projects mature, they need dedicated help in terms of community organizing, legal work, and marketing. That's where the Linux Foundation steps in, by building a technical community and an ecosystem for blockchain development, along with auxiliary support for legal and branding work. Figure 10-1 shows the Linux Foundation umbrella and how the Hyperledger Project is organized under it.

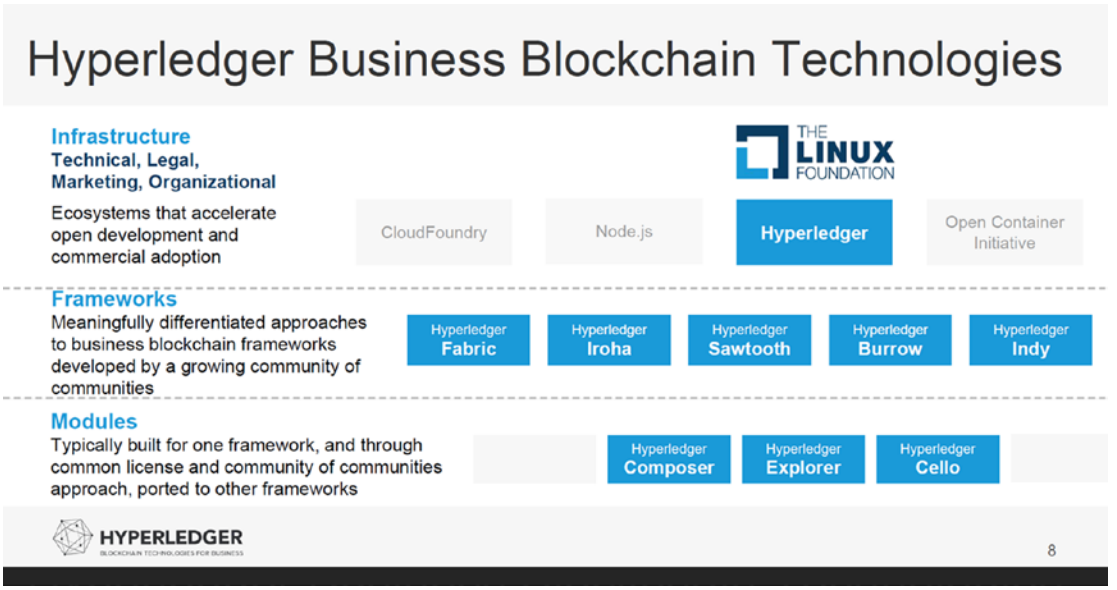


Figure 10-1. *How the various Hypleredger projects and technologies fit under the larger umbrella of the Linux Foundation*

The technical governance in Hyperledger is provided by the Technical Steering Committee (TSC). The TSC is an implementation of Open Governance, whereby major technical design decisions are made by a group of community-elected developers drawn from a pool of active participants. Hyperledger recently had the 2017-2018 TSC elections by following the ABCs of Open Governance: Actively participating contributors are eligible to participate in the election, Bring your nominations for the committee, Cast your vote through a secure channel. In the context of Hyperledger, this mechanism of a developer-led committee to make long lasting design decisions is close to the ideal of a technical meritocracy followed in most open source projects.

Fabric and Sawtooth

Hyperledger Fabric is a modular implementation of the blockchain architecture with pluggable functions including consensus, private membership, and transaction functions. In this section, we take a closer look at three particular features of Fabric: chaincode, types of nodes, and private channels. Fabric blockchain runs smart-contracts in the form of programs called chaincode and the only mechanism of interacting with a chaincode is through transactions. All transactions on the network have to be endorsed, only endorsed transactions can be committed to the blockchain and update the global state. Chaincode can be referenced by two types of transactions:

- *Deploy transactions:* Allow new chaincode to be created, take the program as a parameter. When the transaction has been validated and executed successfully, the chaincode is considered to be installed on the blockchain.
- *Invoke transactions:* Allow a chaincode program to be executed on the blockchain. An invoke transaction calls a specific function from a chaincode that the client requested to execute. It must be noted that invoke transactions only work successfully in the context of a previously deployed program. The result of an invoke transaction is successful execution of chaincode and ultimately, modification of the local/global state with a returned output.

In Fabric, transactions can be made private and to enable this setup, miners/nodes have to be verified on the network. Unlike Bitcoin, here, sacrifice of identity is acceptable because all parites involved in the network have a common goal. To enable private transactions, the nodes have to be updated significantly, adding new features to interact with the various entities on the network. Here, we describe three such fundamental entities:

- *Client:* An entity that acts on behalf of an end-user to create a transaction (deploy or invoke transactions) or broadcast transactions to the network. The client must connect to a peer for establishing communicatation with the blockchain. It often interacts with the ordering service and endorsers.
- *Peer:* A node that commits transactions to the blockchain, maintains state updates and retains a full copy of the ledger. Some nodes have an additional ability to play the role of an endorsing peer. The purpose of an endorsing peer is in the context of endorsing a create-chaincode transaction before it is committed to the blockchain. A chaincode can specify conditions necessary for being endorsed (called endorsement policy) referring to a known set of peers that can validate the create-transaction, except in the case of a deploy transaction (handled by the global or network-wide chaincode endorsement policy). This usually corresponds to receiving signatures from all the endorsers.
- *Ordering service:* A node running the communication service that implements a publish-subscribe messaging routine to perform functions such as network-wide message broadcasting, fail-safe message delivery, and private channels. This ordering service implements the communication protocol in a distributed fashion and gurantees the order of transactions, and the sequences of messages within those transactions is preserved. The communications layer is available to all clients and peers on the network for broadcasting candidate transactions (containing messages) for verification and finalized transactions that can now be appended to the blockchain. Broadcasting is done in a fairly standard manner over shared channels and here's how it works: Clients connect to a channel where peers are listening, and broadcast a message which eventually gets delivered to the appropriate peers.

There are two features of channels (enabled by the ordering services) that we want to discuss here, atomic delivery and partitioning for private channels. The idea behind atomic delivery is to allow message-based communication with order preservation and delivery reliability. In a channel with atomic delivery, transactions are broadcasted and delivered to all connected peers in the same sequence as they arrived. This maintains the consensus of messages contained within those transactions. If the sequence of messages is preserved, once the messages are applied to the blockchain state, the blockchain state update happens in a deterministic fashion (which is desirable). The second feature is partitioning which creates private channels. The ordering service allows mutliple channels to be created, and clients can connect to any channel in a publish-subscribe manner. A client can also broadcast messages under the appropriate permissions to a given channel. Some nodes may create private channels that require permission or prior key-based approval to enter. The ordering service creates channels in such a manner that a client will not be aware of the existence of other channels, even though it can connect to multiple channels. A brief summary of the features available in Fabric is shown below in Figure 10-2.

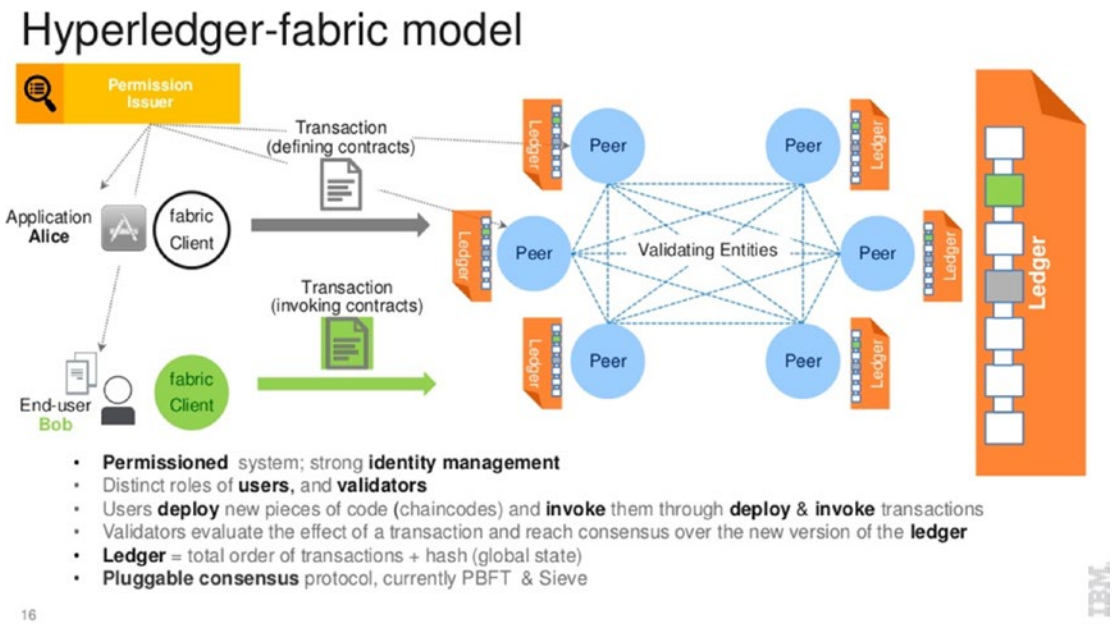

Hyperledger-fabric model
Permission Issuer
Transaction (defining contracts)
Application Alice
fabric Client
Transaction (invoking contracts)
End-user Bob
fabric Client
Peer
Validating Entities
Ledger
Permissioned system; strong identity management
Distinct roles of users, and validators
Users deploy new pieces of code (chaincodes) and invoke them through deploy & invoke transactions
Validators evaluate the effect of a transaction and reach consensus over the new version of the ledger
Ledger = total order of transactions + hash (global state)
Pluggable consensus protocol, currently PBFT & Sieve
16
IBM

Figure 10-2. *A brief summary of the main features offered by Fabric*

Now that we have talked about Fabric in detail, let's briefly discuss Intel's approach to blockchain: Sawtooth. Intel describes their effort as:

> *Sawtooth Lake is a modular platform implemented in Python with C/C++ crypto acceleration, intended for building, deploying, and running distributed ledgers. The platform is modular and extensible, supporting a range of plug-and-play components enabling organizations to rapidly prototype and experiment with various forms of blockchain deployments.*

Sawtooth has a variety of usecases such as compatibility with IoT devices for live data-streams, enterprise-grade customer load, and hardware based security. Here's Intel describing their choice for consensus algorithms:

> *Some use cases may require high transaction rates among a few participants, while others need to support many participants. These diverse environments require different methods of identifying who adds the next block to the chain. We have provided two distinct consensus algorithms to choose from within Sawtooth Lake. For distributed consensus, we have implement a Proof-of-Elapsed-Time (PoET) algorithm, which uses a trusted execution environment to generate fair and efficient leader election to reduce the computation and energy cost while delivering a fair distributed consensus.*

The new consensus algorithms have another advantage to Intel because PoET was designed to run on Intel chips and take advantage of the software guard extensions (SGXs) - an instruction set that creates enclaves to prevent sensitive data access to software-level, and dramatically improve transaction processing times. This hardware acceleration is being used on the consumer side to power a large throughput. The hardware security module in Sawtooth has brought up a very interesting partnership in healthcare with PokitDok announcing Dokchain built on top of Sawtooth. Using SGX allows the blockchain built on top to increase privacy and security, both of which are strong requirements to be HIPPA-compliant. The encryption and security features protection patient privacy make SGX and Sawtooth a very powerful platform for healthcare. Additionally, Dokchain also has adjudcation features whereby once the actors are identified, machine-to-machine payments can happen almost instantaneously. This will significantly reduce the billing cycle for providers and make blockchain-based claims processing more lucrative.

Decision Models: Do You Need a Blockchain?

The hype of blockchain technology has spread like wildfire through the corporate world. All sorts of enterprises and companies have created small blockchain development teams to investigate how they can benefit from using the blockchain. Beneath the hype, there are signs of very serious technological development in the blockchain world, many of which are catalogued in this book. However, we are still in the nascent stages: The reality of business advantages from blockchain-enabled applications is to simplify business processes and reduce friction significantly. In this section, we present three decision models that can help you critically analyze whether integrating a blockchain into your project or company is indeed the appropriate technical solution.

- *IBM model:* This blockchain model highlights how the key features of the blockchain fit into business processes and applications.
- *Birch–Brown–Parulava model:* This model helps you pick between a permissioned and permissionless implementation of a distributed ledger. It was designed by developers at Consult Hyperion.
- *Wüst–Gervais model:* This simplified model, sesigned by two researchers from ETH Zurich, combines the best from both the previous models.

■ **Note** Why did we pick these three models? There are several comprehensive decision trees available online to help a business decide which application scenarios would require a blockchain. In this section, we picked the simplest models to illustrate the more important design principles of integrating a blockchain into existing infrastructure.

Before we dive into the specifics of each model, what general principles are shared by all three? You can use the shared designs as criteria to evaluate your business needs, and create a model specific to your business based on a few simple ideas and questions.

- *Database:* You need to begin by understanding why your business is using a database. Blockchains use a shared database that is visible to all the participants. Are you comfortable with using a shared ledger for your application? The database is constantly updated by transactions that happen over the network. Can your application interface with a database that is constantly updated?
- *Writers:* The database is updated by multiple transactions writing to it. Blockchain is designed to have multiple writers; that is, multiple entities that are generating and verifying the transactions that are published to the ledger.
- *No trust:* Does verifying the identity of writers matter for your application? This could change the ledger implementation that you deploy. The default setting for a blockchain is no trust between parties writing to the ledger. Is that sufficient for your application? Do the parties on your network have competing interests or similar motivations? If they have similar interests, some of the blockchain constructs managing trust can be safely removed.
- *Disintermediation:* Blockchains remove a centralized authority from the network, and the transactions are therefore considered decentralized. These transactions are independently verified and processed by every node on the network. Do you want or need this disintermediation? Given your application-specific use case, are there any significant shortcomings to having a gatekeeper? Good reasons to prefer a blockchain-based database might include lower costs, faster workflows, automatic settlements, or regulatory influences.
- *Transaction interdependence:* A blockchain is best for handling transactions that build on each other, as they are published to the blockchain by writers. The interdependence part refers to transaction clearing. For instance, if A sends funds to B, and then B sends funds to C, the second transaction cannot be cleared until the first one passes through. Blockchain ensures the orderly and rapid processing of transactions published by writers. In other words, a blockchain truly shines when keeping a log of transactions with a long history involving multiple users.

Let's start looking at decision models. The first one is shown in Figure 10-3.

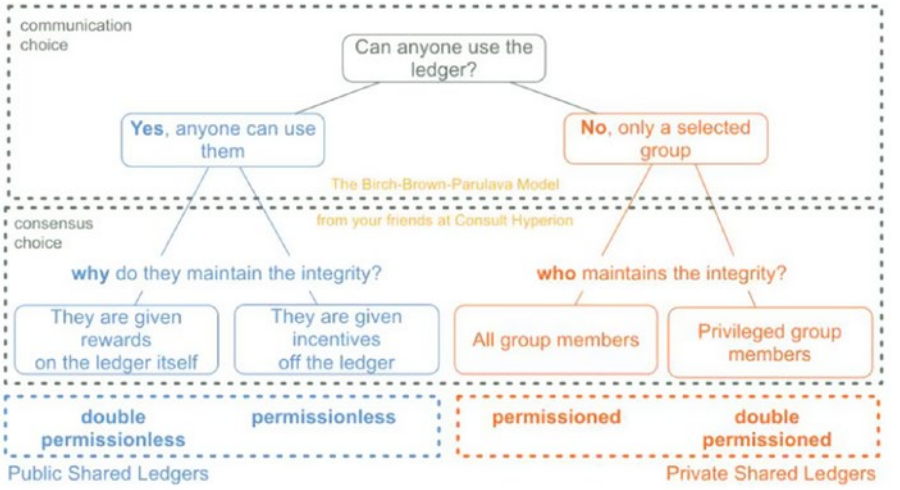


Figure 10-3. *The Birch–Brown–Parulava model*

This model walks a user through making the decision of whether to deploy a permissioned ledger or a permissionless ledger. This could lead to profound changes for deciding the software base, for instance, between Ethereum and Quorum.

The next model is by IBM, shown in Figure 10-4.

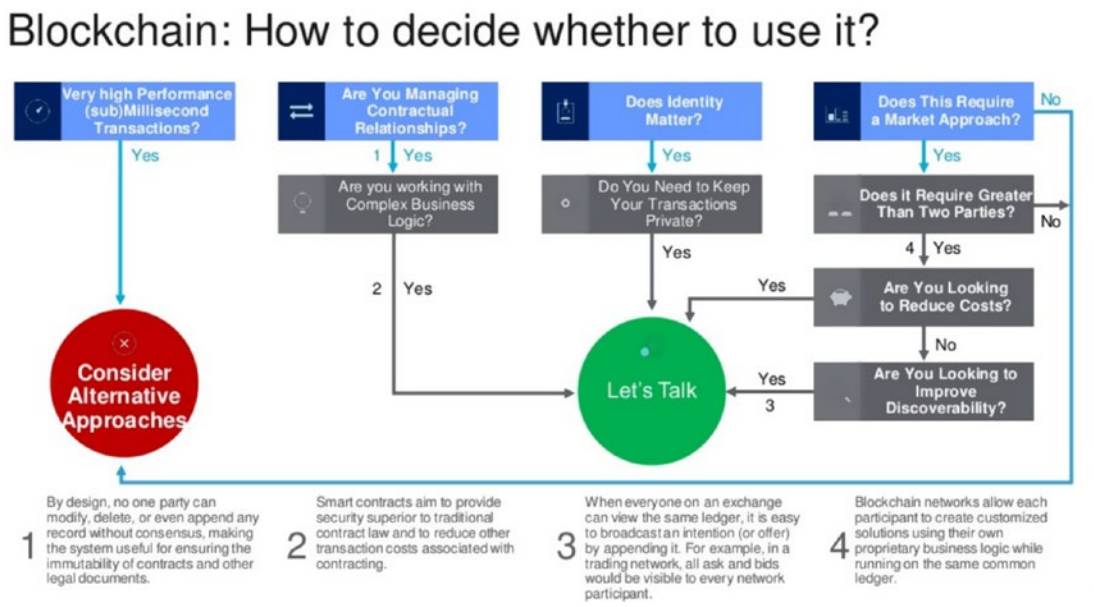

Blockchain: How to decide whether to use it?
Very high Performance (sub)Millisecond Transactions?
Yes
Consider Alternative Approaches
Are You Managing Contractual Relationships?
1 Yes
Are you working with Complex Business Logic?
2 Yes
Does Identity Matter?
Yes
Do You Need to Keep Your Transactions Private?
Yes
Let's Talk
Does This Require a Market Approach?
No
Yes
Does it Require Greater Than Two Parties?
No
4 Yes
Yes
Are You Looking to Reduce Costs?
No
Yes
3
Are You Looking to Improve Discoverability?
1 By design, no one party can modify, delete, or even append any record without consensus, making the system useful for ensuring the immutability of contracts and other legal documents.
2 Smart contracts aim to provide security superior to traditional contract law and to reduce other transaction costs associated with contracting.
3 When everyone on an exchange can view the same ledger, it is easy to broadcast an intention (or offer) by appending it. For example, in a trading network, all ask and bids would be visible to every network participant.
4 Blockchain networks allow each participant to create customized solutions using their own proprietary business logic while running on the same common ledger.

Figure 10-4. *IBM's decision chart for when to use a blockchain*

An important feature to highlight in this flowchart is how certain decisions are associated with features inherent to the blockchain. For instance, contractual relationships can be handled well on the blockchain using smart contracts. This chart can become a decent preliminary check for blockchain compatibility with your project and the features that are enabled by the blockchain.

The final decision model that we consider here is shown in Figure 10-5, which was created by Wüst and Gervais in a paper titled "Do You Need a Blockchain," and for us, it sequentially integrates concepts from both the previous models.

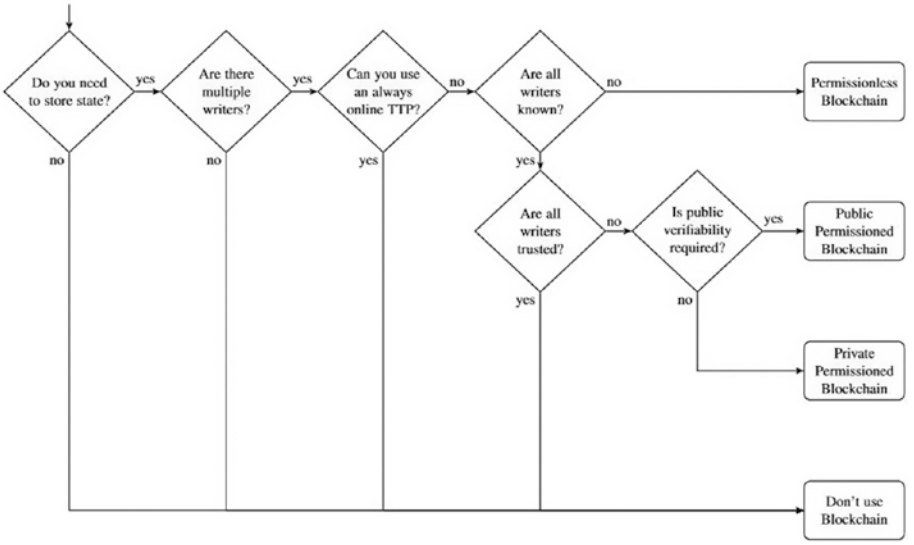


Figure 10-5. *The Wüst–Gervais model*

This model tries to answer for both deployment options and whether you should be considering a blockchain or not. The concepts are similar to the previous models, but the flow is slightly different to allow for additional options.

Rapid Prototyping with Hyperledger Composer

Hyperledger Composer is a suite of high-level application abstractions to model your business. Once you decide that your business might benefit from using a blockchain, you can use Composer to create a blockchain-based prototype of your application. The following is an outline of the process involved in making a prototype:

- Install Hyperledger Composer Tools or try Composer online in the Playground.
- Define Business Network, Assets, and Transactions.
- Implement any transaction processors.
- Test the business network within Composer-UI.
- Deploy the business network to a live Hyperledger Fabric blockchain instance.
- Create a sample application on top of the low-level abstraction.
- Connect other applications to this deployment through RESTful API or Node.js.

The functional output of Composer is called a Business Network Archive (BNA), which can then be deployed to a testnet or a live Fabric blockchain. A business network contains participants that are connected via a role or identity, assets that propagate over the network, transactions that describe asset exchange, contracts that serve as rules that underpin transactions, and finally the ledger that records all transactions. Aside from the modeling components of a business network, the BNA also contains transaction processors, access control lists, and query definitions. A transaction processor (written in JavaScript) implements and enforces business logic on these model elements. The access control lists describe rules that allow fine-tuned control over which participants can have access to what assets in the business network after satisfying certain conditions. The language used to describe these access lists is very sophisticated, and can capture complex ownership statements. Keeping access control separate from transaction processor logic allows the developers to maintain and update the two components rapidly, without any fears of compatibility or breaking functionality.

■ **Note** In this section, our focus is on presenting the fundamentals of Hyperledger Composer so that an interested reader can pick up on Composer's modeling language quicker from the reference material. Designing a full business network model would be beyond the scope of this book.

Let's go through the modeling language and discuss the foundations of Composer in depth here:

- *Assets:* These represent the resources being exchanged between users on the network. Formally, the description scheme used to define assets is done in the order of `keyword-class-identifier-properties-relationships`. Other optional fields could be added here as well, but this general scheme is followed throughout the modeling language. Assets are defined with the keyword `asset` and have a domain-relevant class name associated with them. For instance, a car can be identified by VIN as: `asset Vehicle identified by vin`. The assets have a set of properties that define them. In case of a vehicle, it can be a description stored as a string description. Additionally, the assets can also have relationships to other resources in the modeling language such a user that owns the car, denoted by `--> User owner`. All the assets are stored in an asset registry available to the network.

- *Participants:* These represent the users (or counterparties) in the network. Participants are key to all interactions happening on the network, and they are described the same way as an asset. Participants are represented using the `participant` keyword and they also have a class name relevant to their domain such as `buyer` or `seller`. A full description could be given as `participant Buyer identified by email` and a property of `String firstName`. All the participants are stored in a participant registry, which might not be available to the users of the network.

- *Transactions:* These represent the movement of assets across the network and the life cycle of assets as they are exchanged. Transactions are described in the same way as participants or assets, and a sale offer from a participant could be defined as `transaction Offer identified by transactionid` and with a property of `Double saleprice`.

- *Concepts:* Concepts are abstract classes that do not belong to assets, participants, or transactions. Concepts are often used to add specifiers to an asset or transaction and typically contained by one of the three. For instance, a concept of address can be described as `Concept Address` with properties of `String street` and `String city_default ="Orlando"`. Now an asset can use this concept to specify an address property.

- *Transaction processors:* These represent the business logic to perform state changes and global updates to the blockchain as a result of transactions being broadcast to the network. Essentially, these processors provide an implementation for transactions to act on the model. They are written in a separate js file and use the definitions from the business network model.

  Figure 10-6 shows the BNA visually.

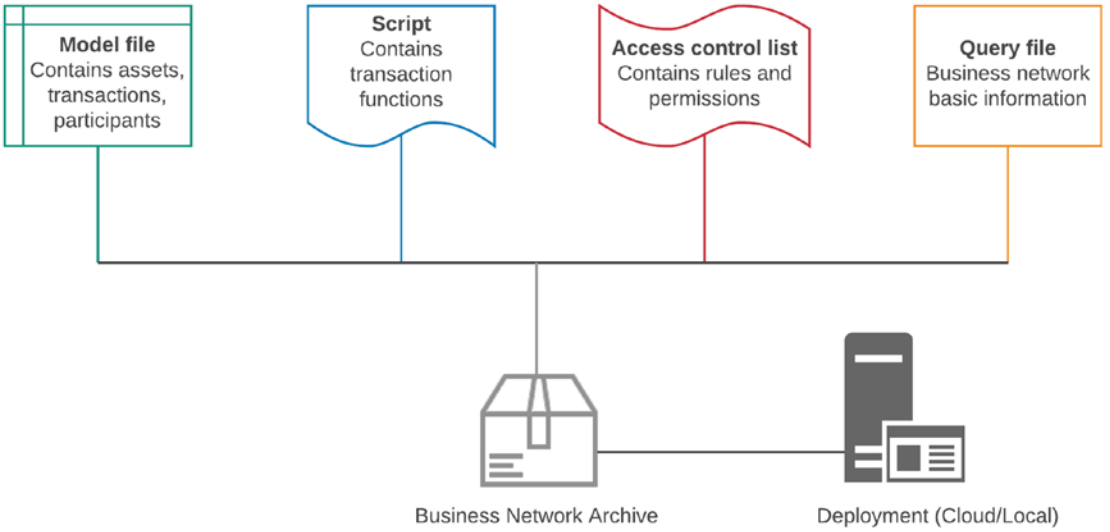


Figure 10-6. *Components of the BNA. Reproduced with permission from the Hyperledger Project.*

Summary

In this chapter, we presented the Hyperledger Project, with a special focus on the currently incubating tools. We began by talking about the five major open source projects currently under Hyperledger, and how the projects are organized under the Linux Foundation umbrella. We then discussed some of the recent developments within Hyperledger, focusing on Fabric and the enterprise features offered in general by Hyperledger. After that, we presented a few decision models to help a project or business decide whether a blockchain would benefit them. Finally, we presented Hyperledger Composer, which is a rapid prototyping tool to help users model their business networks with ease using the modeling language. We concluded our discussion here by talking about components of the modeling language and how they create a business network.

CHAPTER 11

Recent Developments in Blockchain

In this chapter, we focus on three new technologies that have significantly advanced our understanding of blockchain-enabled applications and opened up several new avenues for research. We begin our discussion with EOS, an Ethereum competitor built with an OS inspired architecture and platform-support design philosophy. It uses a new consensus mechanism called delegated proof-of-stake to enable rapid transaction verification and a continuous cycle of voting for delegates that support the network. The message passing protocols implemented in EOS are very advanced, allowing automated response handlers and action triggers on message delivery. They make up most of the smart contract system. There is also support for parallel lockless execution of smart contract instructions across the network, massively reducing latency in communication and state updates.

The second technology we present is a contract-oriented programming language called Chain-core. On the Chain network, all value is concentrated within assets that can be issued and controlled by users. Chain provides developers with a very powerful API to manage and transfer assets in the blockchain. In a similar regard to Aragon, Chain provides a visual interface to manage assets on a blockchain. We do a walkthrough of Chain's graphical interface, covering all the basic features available for asset management. Chain recently announced a new development language to simplify the work of writing contracts for assets called Ivy Playground, and we end the section by talking about the new updates in Ivy.

The final technology that we talk about is Quorum, an enterprise-grade fork of Ethereum built by the Ethereum Enterprise Alliance (EEA) to handle high-volume requirements of businesses using the blockchain. It features numerous upgrades, such as a public–private split on the blockchain (to allow completely private transactions), private smart contracts, and the QuorumChain consensus mechanism. In addition, zero-knowledge proofs will be coming to Quorum in the near future. We end this section with an overview of Quorum given by JPMorgan and EEA.

EOS Blockchain

EOS.IO is a new architectural approach to building a blockchain that resembles an OS. The design considerations allow for core infrastructure that can be scaled with ease for decentralized applications. Here, scalability refers more generally to a set of principles that make blockchain applications competitive with their traditional nonblockchain counterparts. In this section, we focus on the potential for large-scale

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_11

adoption made possible due to three major technical innovations behind EOS: advanced permissions system, a new method for consensus called delegated proof-of-stake (DPoS), and parallel processing in the EOS blockchain. Let's begin our discussion with a few principles of scalability embodied by EOS.

- *User base*: A disruptive EOS blockchain application should be able to support millions of users and user accounts. The underlying architecture needs to be designed such that a distributed ledger can handle accounts as the fundamental unit (as opposed to transactions). Supporting services can provide synchronous modification of accounts and the blockchain global state.
- *Free access*: The services or applications built on a blockchain should not pass on any execution costs to the users. Large nonblockchain services are essentially free and the end user is monetized based on data generated from their usage. To enable widespread adoption, blockchain-based apps would have to eliminate user fees, amortize service costs, and generate revenue from new accounts signing up for the app.
- *Updates and forks*: The integration of new components or features to a service should not require any downtime. Blockchain-based services have to deal with consensus at some level, and disagreements might lead to forks in the chain. These forks create chains with different lengths and are normally resolved very rapidly with the creation of the next block; however, the more serious problems arise with software updates. Bugs in the network should be fixed easily and seamlessly, without the need for a type of hard fork whereby some portion of the network is no longer compatible. The decentralized nature of blockchain services creates a network without a single point of failure and combined with redundancy, it can provide a unique no-downtime experience.
- *Low latency*: The Bitcoin blockchain currently suffers from high latency and incredibly long verification delays. A pragmatic service used by thousands of users is unsustainable with long wait periods, therefore, services on EOS must provide an incredibly rapid verification and confirmation method where applicable. There are some interesting new features such as parallel processing that could alleviate the pain of slow confirmations on transactions.

Let's start with how EOS handles accounts and user permissions. In EOS, all the accounts can be created with unique human-readable references. Additionally, EOS provides a well-defined message passing protocol between accounts with handlers and automation rules. Every account on the network can send structured programmatic messages to other accounts, and define rules and scripts to handle incoming messages when received. The scripts used to handle messages can also send messages in response to a specific incoming action. Each account also has private storage that can be accessed by the message handlers to modify the account state. The message handling scripts along with rules comprise the smart contract system deployed by EOS. The design principles behind EOS smart contracts are analogous to well-researched message passing protocols describing how an OS communicates with peripherals and other hardware components. The contracts will be executed on a virtual machine similar to an EVM. Existing smart contracts written in Solidity or other contract-oriented language can be ported and adapted to work with a container on the EOS blockchain. The contracts will also inherit some EOS-specific functions that allow communication with other EOS components.

■ **Note** Another advanced feature in EOS is the ability to communicate between EOS blockchain and external blockchains compatible with Merkle proofs. This takes place on the client side by using the same fundamental cryptographic principles (proof of message existence and proof of message sequence) to verify messages transferring in and out of the blockchains.

Let's return to understanding how user permissions work. In EOS, the application structure is such that the authentication and permission modules are kept separate from the business logic. This allows developers to design tools specific for permission management and streamline apps to only contain code relevant to app-directed actions. EOS allows the user to define the keys used to sign outgoing messages and the different keys used for each separate service accessed by the user. For instance, a user can have one key to sign messages sent to other users and a different key to access social media accounts. It is possible to provide other accounts permission to act on a user's behalf without assigning any keys. This type of limited access can give someone post-only permissions to your social media (with prior permission), but the posts will retain their unique signature. This is accomplished by a concept called permission mapping that we discuss next. A key idea in permission mapping is multiuser control of a single account. This is considered a high-level security measure for decentralized apps, and in EOS, it is implemented to reduce the risk of password or currency theft.

■ **Note** EOS can be considered a generalized evolution of the blockchain-based social network called Steem created by Dan Larimer. Steem was built on the Graphene blockchain and permission mapping was originally implemented as a feature, but it was very limited. Whereas Steem was designed as a decentralized social platform, EOS is an application development platform with a Turing-complete blockchain analogous to Ethereum.

To understand permission mapping, let's look at an example involving three user roles: the account owner, an active user, and a social media user. The social media permission allows only specific actions by the user such as voting and posting. On the other hand, active user permission allows almost any action except for removing the owner. The owner can perform all available actions, including withdrawal. In this example, permission mapping refers to an account owner defining roles and mapping the associated permissions to other accounts. EOS generalizes this example by allowing custom roles, custom actions that can be mapped to different accounts. For instance, an account holder can map the permissions for a social media app to a Friend user group. Now any account added to the permission group (by the account holder) instantly has access to the social media app. They can post, vote, and perform any other actions available to the permission group but their unique keys would be recorded. Therefore, it will always be possible to identify which user performed an action. The concept of permission mapping is illustrated in Figure 11-1.

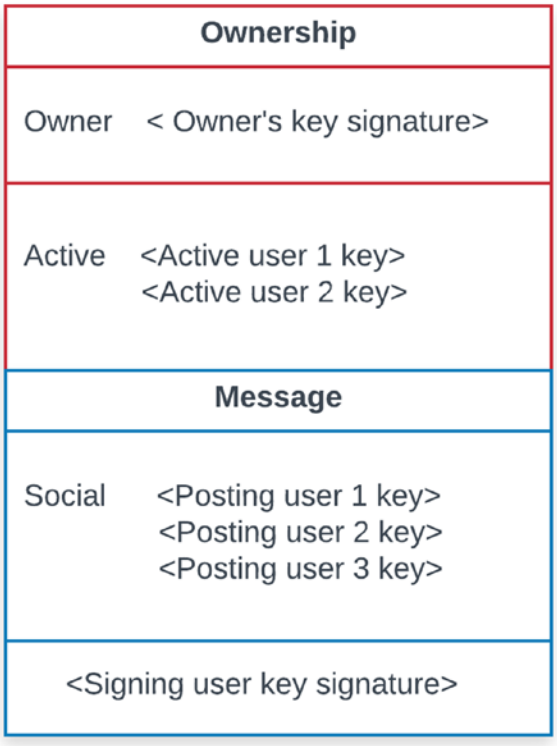


Figure 11-1. *Structure of custom permission mapping*

We can divide this representation into two components, the ownership and the message itself. The ownership portion contains keys from the account holder, who can perform all account functions including withdrawal of tokens. It also contains keys from active users that act on the behalf of the owner. The second portion of this structure is the message, and here we continue to use an example of social media. There are three social users mapped to this account by the owner and each user has a unique key. When a message is posted to the social media, the posting user signs the message with his or her own unique key, so any actions performed by the mapped users can be tracked. The message module here can be replaced by other applications, with the account holder mapping custom permissions to the app, providing limited functionality to the mapped users.

Delegated Proof-of-Stake

EOS uses a new consensus mechanism called DPoS. It is a variant of the proof-of-stake (PoS) algorithm with a focus on efficiency and fair access, giving smaller wallets or accounts a candid chance to produce blocks and earn the reward. The difference between a traditional PoS algorithm and DPoS can be compared to direct vs. representative democracy. In traditional PoS algorithms, any user with a wallet can participate in the process of validating transactions, forming consensus and earning a part of the mining reward. However,

it might be unprofitable for users with small wallets, or a smaller stake. In DPoS, every wallet across the network with any amount of coins can vote for delegates who perform the functions of validation and appending blocks to the blockchain. Ultimately, there are two main players in a DPoS network.

- *Stakeholders*: These entities have an account in the EOS network that holds a wallet with a balance. Having tokens allows you to vote for block producers.
- *Block producers*: Also known as delegates, these entities drive the consensus mechanism of DPoS. Block producers are the DPoS equivalent of traditional miners. They validate transactions on the network, sign blocks, and earn a reward for adding a block to the blockchain.

Let's briefly review the mechanism of consensus and the need for global state updates before diving deep into DPoS. Recall that a blockchain can be modeled as a state machine with a consistent history of updates. This state machine is acted on by transactions and consensus is the process of network-wide agreement on the order in which transactions will update the global state. The updates occur at specific increments as new transactions are included in blocks, and the rapid frequency of updates filters out invalid or double spend transactions. The goal of DPoS is to make block production evenly distributed among the most people and to have a democratic, fair process that elects the block producers. Regular PoS networks require full consensus to validate, whereas in a DPoS system a limited number of delegated nodes (delegates) can probably validate transactions and reach consensus.

■ **Note** In the current schedule, EOS allows blocks to be produced every three seconds by a singular block producer. By default, the blocks are produced in multiple series, with each series containing 21 block producers.

The DPoS algorithm has two major operational roles: electing a group of block producers and scheduling production of blocks. Each account on the network is allowed at least one vote per delegate (miner) in a process known as approval voting. However, accounts with a larger stake in the network follow the principle of one vote per delegate per token, and can vote for more than one delegate. The block producers (delegates) are considered a minimally trusted party on the blockchain, being elected through a real-time voting process. The delegates can take turns (in a series) adding transactions to a block and signing blocks to be added to the blockchain. As such, the use of a trusted party prohibits malicious parties adding invalid transactions to the blockchain. It should be noted that delegates don't have the ability to change transaction or block details; they are only able to either add or not add a transaction to a block. This group of delegates are randomly assigned for block production based on some variation of a weighted fair queueing algorithm. The goal is to efficiently schedule block producers based on two criteria: the number of votes they received and the time a delegate has been waiting to produce a block. Delegates with more votes are more likely to be scheduled next to produce a block. On the other hand, delegates acting badly can be voted out quickly without affecting any significant transaction volume. Let's summarize the main roles of delegates.

- *Block production*: The most important function of delegates is to create blocks at a fixed schedule of every three seconds. This involves validating transactions and signing blocks across multiple series.
- *Network support*: Each time delegates produce a block, they are paid a reward for forging the block. The pay rate is set by the stakeholders, but some delegates might accept a lower pay rate to get more votes from the network. Additionally, instead of taking lower pay rates, delegates might offer additional services, such as marketing and legal work, to the stakeholders.

Figure 11-2 shows representative block production used in DPoS consensus algorithms.

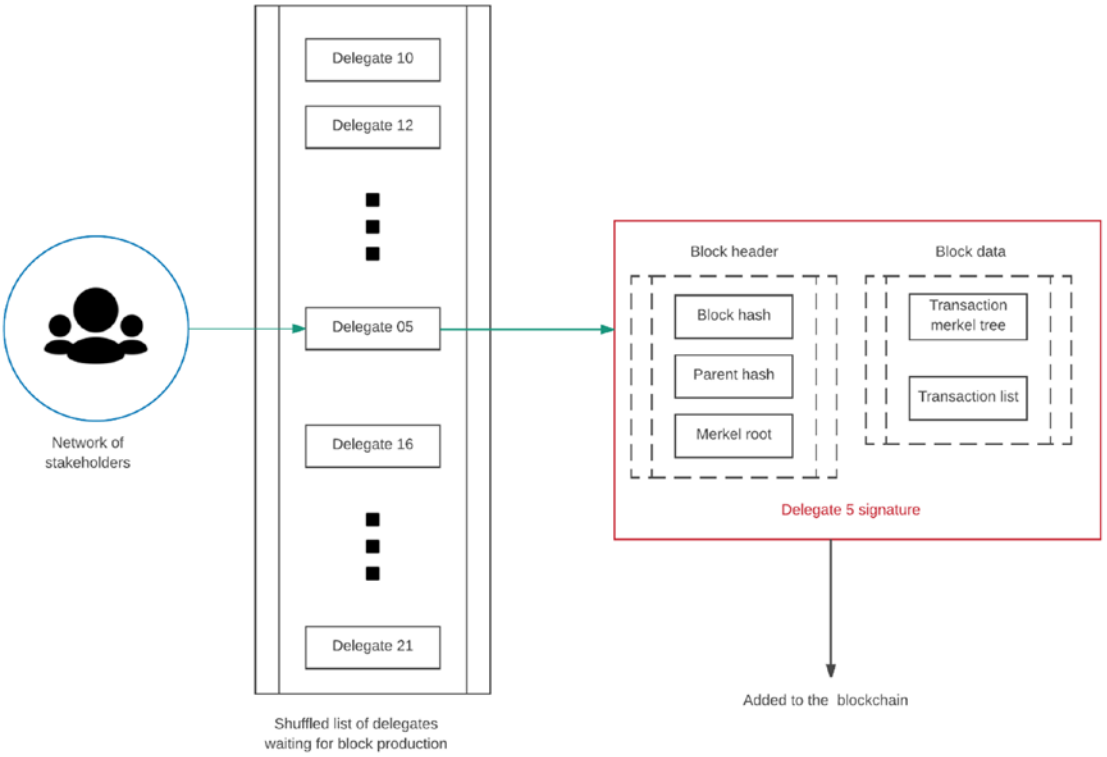


Figure 11-2. *Block production in DPoS*

As mentioned previously, stakeholders on the network vote and elect delegates, who in turn validate transactions and produce blocks. From the highest voted delegates, 21 are elected to participate in multiple series and produce blocks. The delegates are assigned to a schedule for producing blocks, and this assignment is made using a round-robin algorithm. A block is considered invalid if a block producer forged outside of their scheduled time slot. After a block has been signed by the delegate, the final block can be appended to the blockchain, as seen on the right side of Figure 11-2. If a block producer fails to be on schedule, the next delegate's block will be larger and include the previous transactions. At the same time, a block producer that fails repeatedly will be voted out of the series.

■ **Tip** It is interesting to note that DPoS blockchains do not have a tendency to fork. This is because block producers cooperate to produce blocks, rather than competing in a PoW system. In the event that a fork does occur, the consensus automatically switches to the longest chain.

Parallel Execution

In Ethereum, execution of instructions (e.g., from a smart contract) is done in a deterministic and synchronous manner across the network. EOS offers a very interesting upgrade for parallel execution of applications without the use of any locking primitives. In this case, there must be a mechanism inherent to the blockchain ensuring that each account only updates its own database. Incoming instructions update the account state; therefore, they must be applied sequentially such that the n + 1 instruction updates the account only after nth instruction. There is an exception to this general rule in the case of pass–fail binary instructions. Some account handlers can process binary instructions as read-only and reply back without any changes to their internal state.

Fundamentally, parallel execution in EOS relies on messages generated from different sources within an account, delivered through independent threads, so they can be executed in parallel. Ultimately, the final state of an account only depends on the messages delivered to it. A block producer organizes the delivery of messages to independent threads. Even though a block producer is on schedule for validating blocks, message delivery is done on a more rapid and custom schedule to take advantage of parallelism. Often the source of these messages is scripts running on the blockchain or automatic account handlers sending messages. Due to the parallel nature of EOS, when a message is generated, it doesn't get delivered immediately. Instead, there is a period of delay between message generation and delivery referred to as latency. The rationale behind introducing latency is that instantaneous delivery could interfere with a receiver already modifying its internal state due to a previous message. Lockless messaging is achieved by scheduling the message to be delivered in the following cycle.

So what is a cycle? Under normal circumstances, to avoid message collision, an account would have to wait until the next block to send a second message or to receive a reply. A block is created every three seconds, therefore accounts can expect at least a three-second wait between sending more messages. To remove this wait time between messages, EOS divides a block into multiple cycles and further subdivides those cycles as follows:

- Each cycle is divided into threads.
- Each thread contains a list of transactions.
- Each transaction contains a set of messages that will be delivered.

Figure 11-3 shows the subdivisions and structure of a cycle within a block. Block producers keep adding new cycles to a block until a time limit or no new transactions have been generated.

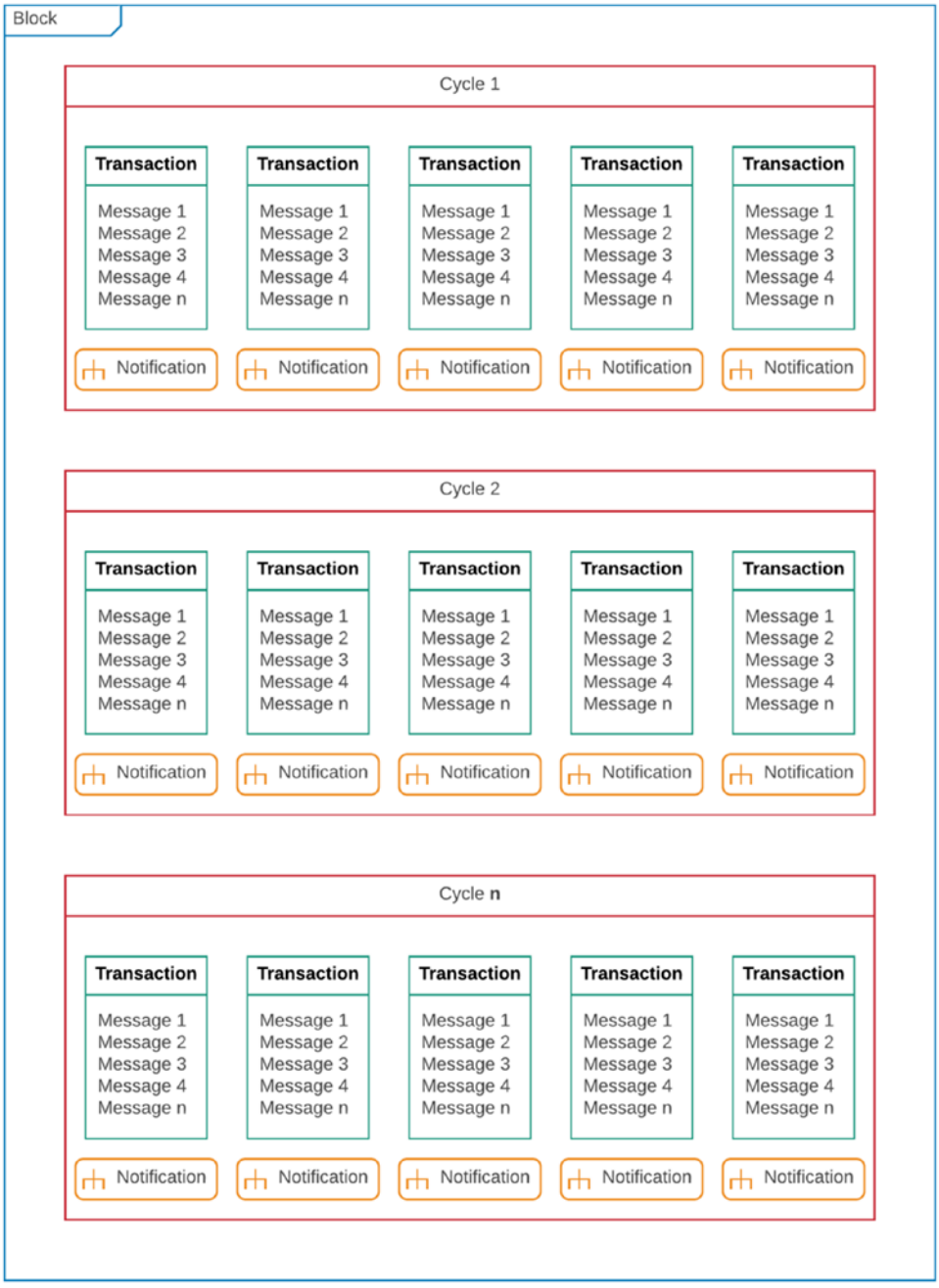


Figure 11-3. *Block structure for messages in EOS*

Each block is divided into a cycle that forms the basis of rapid communication within a single block. If no two transactions within a block modify the same account, all the transactions from one cycle can be executed in parallel. For each transaction that takes place between users, any automated handlers preset by the receiver are notified of incoming instructions.

Transactions generated during the time window of one cycle can also be added to a subsequent cycle or the next block. Once a block is finalized, we can ask this question: Given a block cycle, are there two threads that contain transactions that modify the same account? If not, then any given block can be processed by executing all the threads in parallel. An exchange application developer runs full nodes for the purpose of displaying the exchange state to its users. This exchange application has no need for the state associated with social media applications. EOS.IO software allows any full node to pick any subset of applications to run. Messages delivered to other applications are safely ignored because an application's state is derived entirely from the messages that are delivered to it. There are two important take-home points that need to be reiterated from this section:

- An application's internal state is derived from the messages delivered specifically to that app.
- Any changes to an account's internal state happen via messages passed between accounts and included in transactions on the blockchain.

Dan Larimer commented on the parallel nature of EOS as a massive advantage for building applications on the blockchain:

> *Parallel processing capabilities bring continuous scalability and a reliable high performance foundation for your applications. Existing single threaded capabilities force every application to share the capacity and performance of one single-threaded blockchain, create hard scaling limits, and eventually suffer from network congestion that may result in platform-wide downtime.*

■ **Note** EOS.IO recently had an ICO to raise funds for development, and CryptoPortfolio reviewed the ICO, making a few points worth considering for us. The first point is that profit generation comes from the value appreciation of EOS tokens. The more users and developers, the more valuation. The second point is that there are no transaction fees on EOS, so accounts and DApps need tokens to work. Holding certain amount of tokens is like having bought a percentage in EOS computational power. The third point is there is going to be free usage of the platform; you only need tokens in your balance to operate on this blockchain. Finally, the last point is that users demand fast response from services they use, so EOS will try to make transaction confirmation in less than 1.5 seconds.

Scheduling

The last topic that we want to tackle in this section is that of best effort scheduling. Recall from our previous discussion of computational marketplaces that a virtual machine (EVM) on the blockchain determines the cost of execution for the instructions contained in a contract. The customer requesting a given task paid for each step of the instruction with gas, and if the number of steps exceeded the EVM limit, the task would not execute. In EOS, the delegates take over the role of making global decisions about eligibility to

execute, instead of a virtual machine. The block producer (delegates) makes a subjective measurement of the complexity (number of instruction steps) and time required to process the instructions contained within a transaction and deliver a message. This subjective measurement of processing costs is analogous to gas in an Ethereum system. A delegate-based instruction counting and cost-determining mechanism simplifies the addition of new features and optimizations to the blockchain, without any concern about breaking the cost calculations. It also allows us to categorize the network functionally: blockchain and virtual machines for execution and nodes or block producers for calculating resource allocation and management.

Each block producer calculates resource utilization using its customizations to the EOS algorithms. Across the network, each step has an execution time limit and is charged a fixed bandwidth-usage cost regardless of whether the computation took a millisecond or the full time limit. A crucial point to note is that in EOS all resource usage constraints are subjective and ultimately enforced by block producers. As a collective, the block producers have agreed on the following criteria on the EOS network:

- For a block producer, there are two options available: include a transaction (a set of instructions) or reject it from being included in a block.
- If a block producer concludes that a given account has consumed an unreasonable amount of the computational bandwidth, they simply reject transactions from that account when producing their own block.
- As long as one block producer considers a transaction valid, and under the resource usage limits, all other block producers will also accept it. However, this transaction will take longer to confirm on the network.
- If a block producer includes transactions that are well beyond the limits of resources, another producer might reject the block and a third producer will be needed as the tie-breaker.

There would be a delay in propagation of the tie-breaker block, giving the stakeholders and other delegates ample time to pinpoint the source of resource violation. Over time, if a malicious block producer keeps approving blocks that are over the limits, that producer will be voted out from the list of delegates. The block producers perform the role of a best-effort scheduler in EOS: fair scheduling of tasks for maximum throughput or rejecting tasks from being executed on the network.

■ **Note** An initial release of EOS containing the SDK has been made available on EOS GitHub. This release is called Dawn 1.0 and according to the release announcement, it contains a standalone eosd node that produces blocks and adds them to the blockchain, a command line client called eosc, a client wallet and a utility to create a local testnet. It also contains an app for smart contract developers to write contracts.

Chain Core

Chain is a contract-oriented programming language designed to manage assets on a blockchain. The following is an excellent summary of the Chain platform by its team:

> *The key characteristic that makes a Chain blockchain a shared ledger, rather than simply a distributed database, is that it is designed to track and control value—i.e., units of one or more assets. Unlike data, value is scarce: once issued, it cannot be copied, only transferred.*
>
> *We gain new capabilities when the underlying format of a financial instrument becomes a cryptographically-issued digital asset, one of the most notable of which is the ability to write programs that control the asset. These programs are known as contracts.*

> *All value on a Chain blockchain is secured by such contracts. Each contract controls, or locks, some specific value—i.e., some number of units of an asset—and enforces the conditions that must be satisfied before that value can be unlocked. From the simplest custodial account to the most complex derivative instrument, contracts define the rules for asset movement on a Chain blockchain network. And Ivy lets you compose those rules.*
>
> *When designing Ivy we had a single goal: making it easy to write safe, secure, and simple programs to control the movement of value on a blockchain. Nothing more, nothing less.*

At Consensus 2017, a major blockchain development conference, Chain announced a set of more sophisticated upgrades to the core programming language making it easier to write contracts that manage assets. Ivy Playground was the result of this effort to make the developer experience as smooth as possible. A core concept that Chain and Ivy have embraced in designing contracts is to limit the amount of information in the blockchain to only state-updating logic and consensus-maintaining variables. This design rationale was explained by the Chain team as follows:

> *As much as possible, the business logic of blockchain solutions should be executed off-chain, in the "application layer"—programs written in traditional languages, that interact with Chain Core through its SDK. The only logic that needs to be part of the "smart contract" layer is the set of rules that secure and transfer assets.*
>
> *Some blockchain platforms are marketed as workflow automation tools, as a global database for shared state, or as a platform for building software-as-a-service applications. We believe that these goals make the wrong trade-off, by effectively shifting computation and logic from traditional servers, where computation is cheap and information is private, to the blockchain, where every computation must be redundantly executed on every node in the network.*

The key to understanding Chain's design is that traditional services cannot be transitioned to a blockchain; instead they need to be redesigned for a new architecture. For starters, additional parameteres involving consensus are introduced in an application. The programmatic routines need to distinguish between the business logic that should remain outside of the blockchain and state variables that need to be updated on the blockchain. Therefore, new design decisions are needed. Additionally, traditional services have some inherent redundancies that are normally not a problem for servers in a data center, but become amplified in a decentralized network where each node must update the state of accounts in the same order for every step of computation (replicated state machines). To illustrate how Chain works, we do a walkthrough of Chain Core with Ivy. The main objectives of this walkthrough are to understand how to create hardware security module (HSM) keys, how to create accounts, and how to trade assets. The layout will be very similar to Aragon, which we covered previously. Begin by downloading the Chain Core with Ivy Playground (`https://chain.com/docs/1.2/ivy-playground/install`) from the Chain web site. Once it is installed, begin by configuring Chain Core, as shown in Figure 11-4.

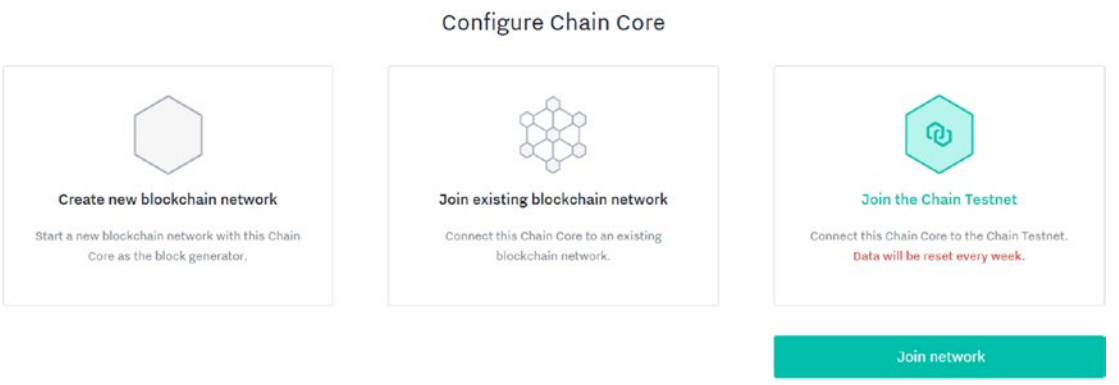

Configure Chain Core
Create new blockchain network
Start a new blockchain network with this Chain Core as the block generator.
Join existing blockchain network
Connect this Chain Core to an existing blockchain network.
Join the Chain Testnet
Connect this Chain Core to the Chain Testnet.
Data will be reset every week.
Join network

Figure 11-4. *Configuring Chain Core*

For our walkthrough, we will be joining a testnet that behaves like a blockchain, but all the data is reset weekly. The testnet helps in understanding how to interact with a blockchain properly and how to manage assets in Chain.

Click Join Network to open the dashboard of Chain Core. Here, you are presented with an option to perform a tutorial. We cover the basics of that tutorial here, adding more information about the main concepts in Chain. To the left of the dashboard is a navigation menu, shown in Figure 11-5, that we will be using frequently throughout the walkthrough. The navigation menu is split into three parts: The top part deals with data related to accounts, assets, and transactions. The middle part deals with private–public keys required to interact with the blockchain. The bottom part provides some developer services, and we will be talking about Ivy Playground later in our walkthrough. To begin interacting with the testnet, we first need to create keys. Chain uses HSM-backed key management and this key becomes your gateway to the network. To create a key, go to the navigation menu and click MockHSM under Services.

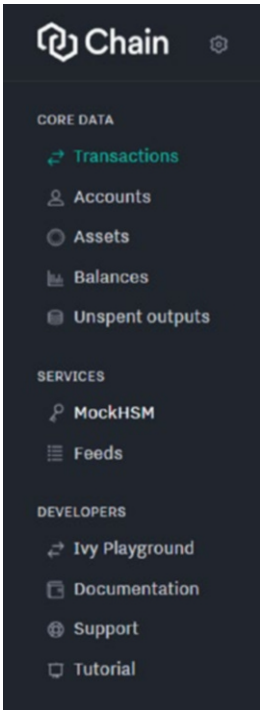


Figure 11-5. *Naviation menu layout on Chain dashboard*

The navigation menu is split into three pieces: Core Data, which includes accounts, assets, and transactions; Services, where we can create HSM keys; and finally Developers, which includes a few developer tools. We will be using the MockHSM tab to create a key pair.

Why do we need a HSM key pair? This key pair will be used to create new classes of assets, issue new assets, and create new accounts. The default view for the Chain dashboard is that of transactions. Click MockHSM and you will see a message saying, "There are no MockHSM keys," which stands to reason, given that we haven't made any keys. Let's start by clicking New MockHSM keys. You should see the screen shown in Figure 11-6. Type an alias for the key and that name will be used throughout Chain.

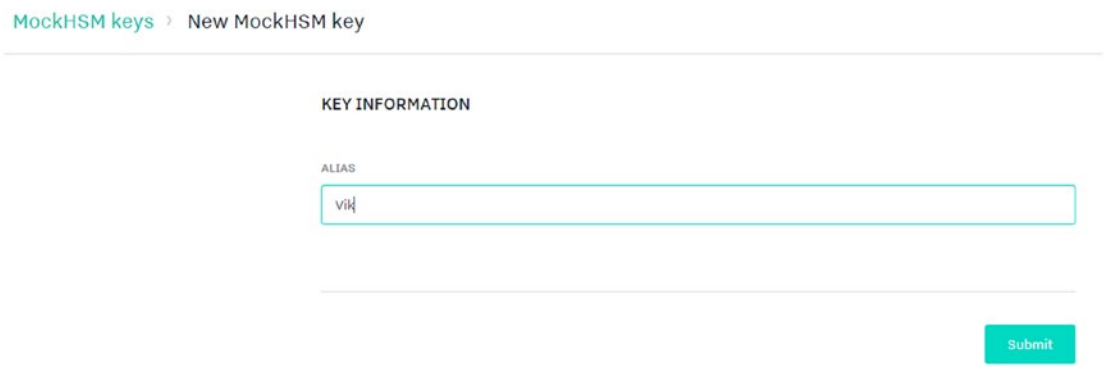


Figure 11-6. *Creating a new HSM key. On the testnet, the key data will be erased in a week*

After clicking Submit, you should see a confirmation for the key that you just created, shown in Figure 11-7.

Figure 11-7. *Confirmation from key creation with an alias Vik*

Now that we have a key pair, we can use it to create assets. Click the Assets tab under Core Data and you should see some default assets (at the time of writing, Chain Core had two defaults), shown in Figure 11-8.

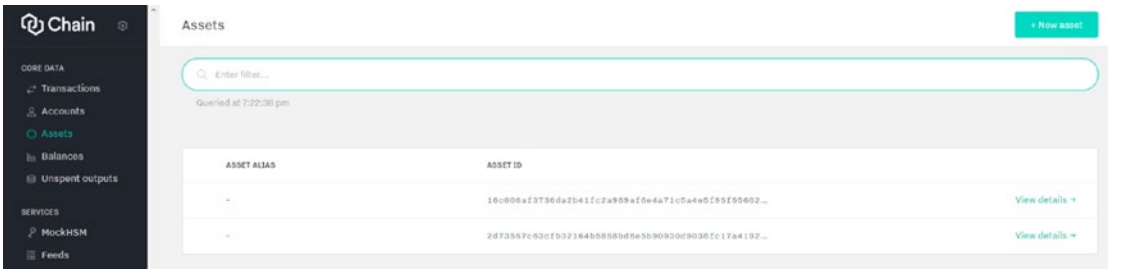


Figure 11-8. *Default assets available in Chain. We will be creating new assets by clicking New Asset at the top-right corner of the screen.*

You should see the New Asset screen shown in Figure 11-9.

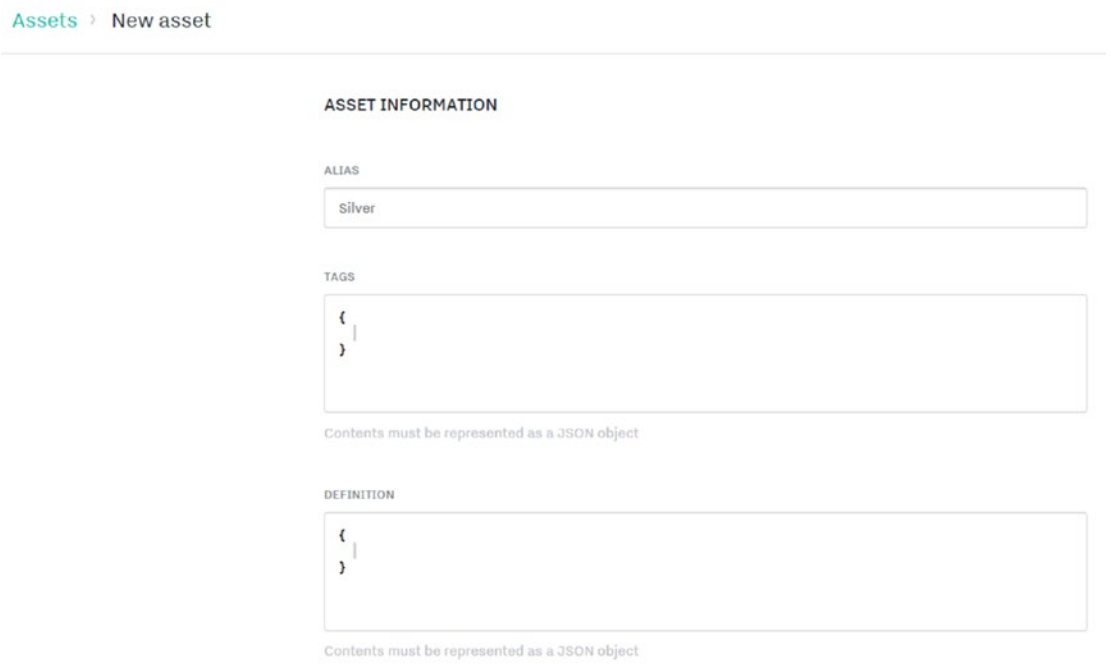


Figure 11-9. *Asset creation and signing*

The process requires two steps. First is inputting the asset information, including any identifiers or definitions assigned to an asset in JSON format. The second step is signing the asset with your public key. Here, we are naming the asset Silver.

Once you have completed the asset information, you need to scroll down and sign the asset with the HSM key that we just created. Figure 11-10 shows the signing process. The Chain documentation provides an example of why a user might want multikey authentication (shown in Figure 11-10):

> *In simple cases, an asset or an account will define a single key required for issuance or for transfers. But it's possible to define multiple keys for different usage patterns or to achieve different levels of security. For example, a high-value asset may be defined with two signing keys, requiring two separate parties to sign each issuance transaction. A joint account may also be defined with two signing keys, requiring only one, from either party, to sign each transfer. The threshold number of signatures required is called a quorum.*

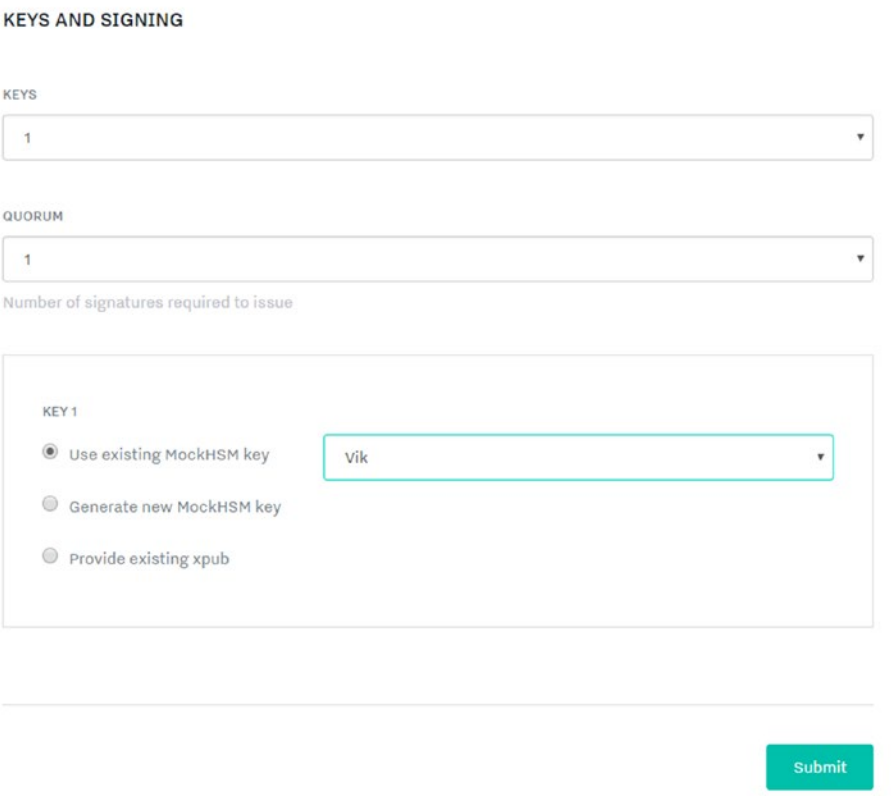


Figure 11-10. *Signing an asset*

There are a few parameters to consider here. By default, a new asset can be created with signature from one key. The number of signatures required to approve the new asset are referred to as Quorum, and we use the defaults here. Finally, we have to sign the asset using the key we created. After you click Submit, you should see a confirmation screen with the Silver asset created on the testnet.

Now that we have an asset, we need to issue it. In Chain, all assets are managed and controlled by users; therefore, the asset we issue would have to be controlled by an account. We don't yet have any accounts on the network, so let's take care of that by clicking Accounts on the navigation menu. This brings us to the account view. There might be some sample accounts on the testnet such as Alice or Bob, but in the right corner of the Accounts screen is the option to create a new account. Click that to open an interface very similar to the one used for creating an asset. In fact, most of the steps are exactly the same here, such as filling out the account information, and then providing an HSM key to sign it. The account creation view is presented in Figure 11-11.

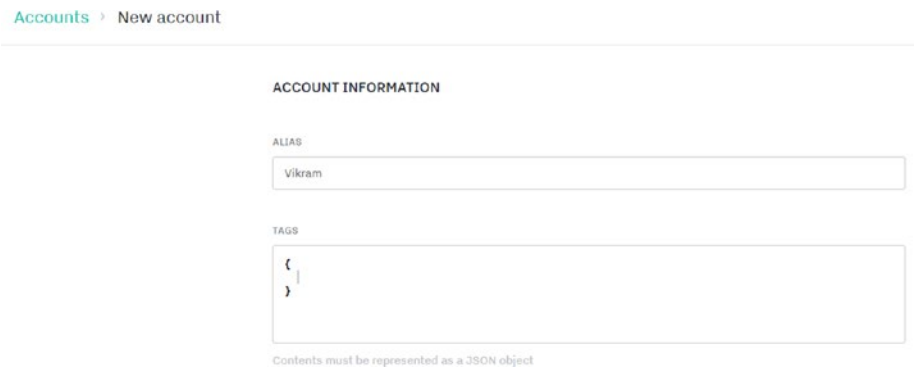


Figure 11-11. *Creating a new account*

There are two steps for creating an account: providing account information and key signing. The first step is shown in Figure 11-11 and the second step is exactly the same as signing an asset.

Once you have provided the account information, sign the account using your HSM key. A confirmation screen will open for the account that you just created. Now, to illustrate trading between accounts, we need two accounts. Create a second account with the alias Team Rocket, and the final result should look like Figure 11-12.

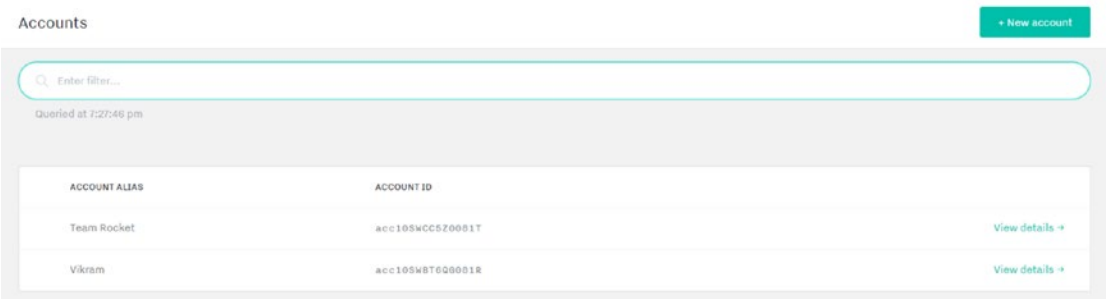


Figure 11-12. *Account summary. Two accounts on our testnet will allow us to show trading, locking, and unlocking.*

Now that we have two accounts, we can start to trade assets between them. It should be noted that in Chain Core, assets can only have four actions performed on them: An asset can be issued, spent, transferred (controlled), or removed (retired). Usually, a combination of these actions allows for transactions between users. We look at two types of transactions here; one is simply issuing new asset to an account, and the second one involves trading that asset to the second account we created. Finally, we end the walkthrough by looking at how these actions have evolved in the Ivy Playground. To begin, navigate to the Transactions tab, and click New Transaction. This should take you to the New Transaction screen shown in Figure 11-13.

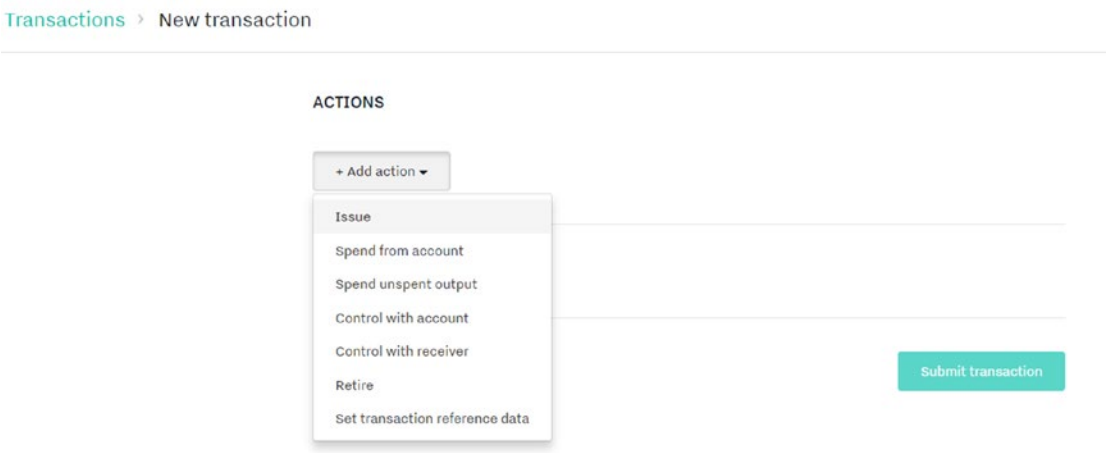


Figure 11-13. *New Transaction screen*

A transaction can only be carried out in terms of one of the four actions we mentioned earlier. Each action has a set of associated parameters that need to be given to complete a transaction. Here, we want to issue a new asset, so we select Issue.

Once you slect Issue, another screen opens, asking you the parameters of the asset that you want to issue, as shown in Figure 11-14.

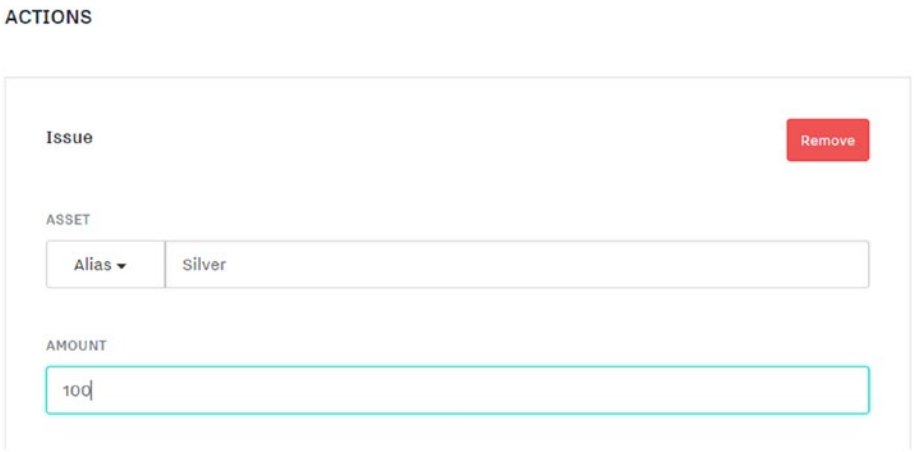


Figure 11-14. *Creating an asset. The issue action first asks for the name of the asset we want to create and finally the amount that we want to issue.*

Recall that in Chain, assets cannot just be issued in a void. Creation of a new asset is coupled with control of the asset by an account. Once an asset has been issued, we will assign it to the first account we created. Figure 11-15 shows the control of an asset being passed to the new account.

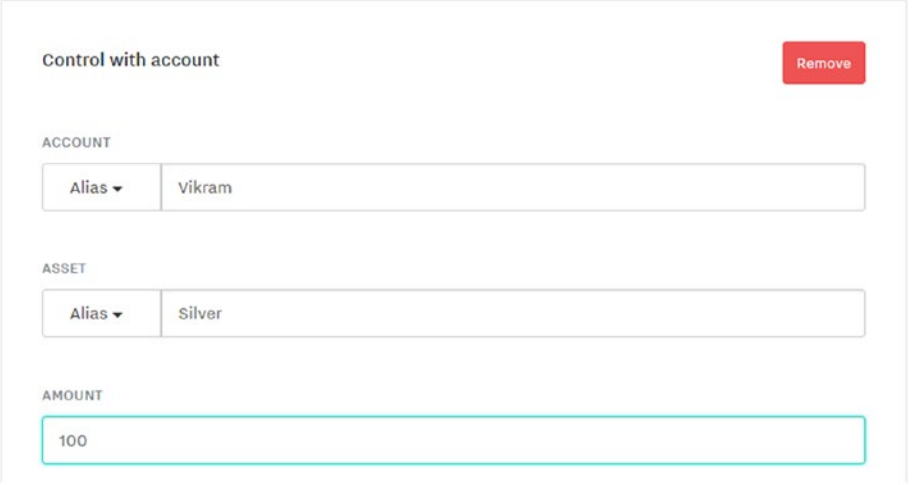


Figure 11-15. *Assigning an asset to an account*

This is also referred to as control with account in Chain. The 100 units of Silver are transferred to the first account that we created.

Once the assignment is filled out, you can submit this transaction. It will be added to the testnet, and the confirmation is shown in Figure 11-16.

Figure 11-16. *Transaction verification*

This summary shows 100 Silver assets issued and then assigned to the Vikram account. Put differently, the assets were issued and then controlled by the first account we created.

This is the simplest transaction we can do on Chain. Now let's move to the next transaction involving a trade between two users. This time around, we will have to use the spend action to make assets available, and control action to transfer them. Go back to the Transactions tab from the navigation menu, and click New Transaction. This time, instead of using the issue action, select Spend From Account, as shown in Figure 11-17.

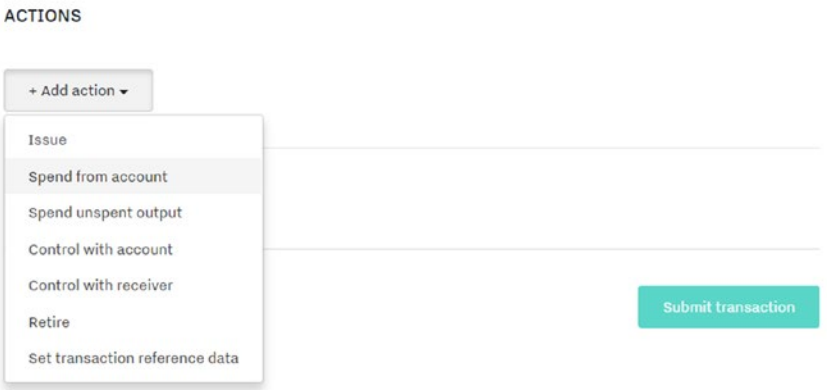


Figure 11-17. *Use the Spend From Account action to initiate a trade*

To start the trade, select Spend From Account. Next, specify how much Silver you want to transfer (Figure 11-18). This first step only makes the asset available; it still needs to be claimed.

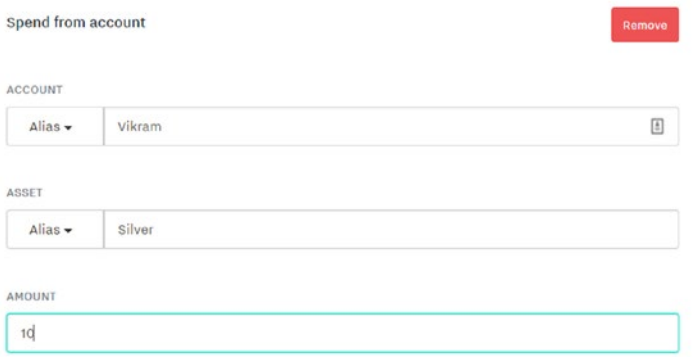


Figure 11-18. *Making 10 units of Silver available for trade from the Vikram account*

Now that Silver is available, let's switch ownership from the first account to the second, as shown in Figure 11-19.

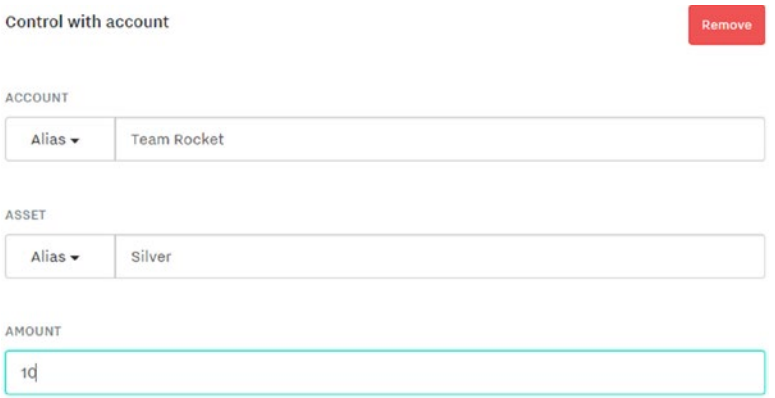


Figure 11-19. *Transferring ownership from one account to the second (Team Rocket in this case)*

Just as we did while issuing assets, trading assets involves one account making assets available, and the second account taking control of the freed assets.

To confirm that the transaction took place, we can go back to the Transactions view, and then look at the most recent transaction. Clicking it will expand the transaction (see Figure 11-20).

Figure 11-20. *Summary of the most recent transaction*

Notice that 10 assets were spent by the first account (Vikram) and 10 assets were then controlled by the second account (Team Rocket), completing our trade.

So far, we have discussed two transactions that are key to Chain Core. There are a few more options available, such as retiring assets, which simply means taking a given amount of assets out of circulation. This has the effect of increasing scarcity for the network. Now that we understand the concept of actions, what more can we do in Chain? Writing sophisticated scripts to manage and move assets in Chain becomes a little complicated. Ivy Playground was introduced to develop a formal scripting language around locking and unlocking assets on the blockchain. Developers can add conditions to a script and write more complex contracts to secure value and assets. The main objective of Ivy is to lock the value of an asset with a set of clauses that describe conditions under which the asset can be unlocked. The Chain team talks about how contracts are different in Chain as compared to other distributed computational platforms such as Ethereum:

> *Contracts on a Chain blockchain are immutable. A contract cannot be updated, only spent—i.e., removed from the blockchain and replaced with one or more new contracts that lock the same value.*
>
> *This immutable state model distinguishes our approach from that of (for example) Ethereum and Fabric. In Ethereum, contracts can send messages and value to each other and can update their own state in response to these messages.*
>
> *Most current smart contract approaches provide the freedom to write programs that execute arbitrary logic on a blockchain, but make it essentially impossible to provide the kind of safety assurances that Ivy is designed to give. While other platforms have the goal of making it easy to build anything, Ivy's goal is to make it difficult to build anything wrong.*

Let's finish our walkthrough by looking at a simple example of using a public key pair to lock assets and then unlock them. To begin, click Ivy Playground on the navigation menu. This should bring you to the Playground with a default contract loaded that locks assets using a key. The idea is very straightforward: You choose the values to lock; for instance, in the case of an asset, we can lock the units of silver, and then specify the account the asset is coming from. Then we provide arguments to the contract, and here, the only argument is that the asset is locked using a key pair. This idea is shown in Figure 11-21. After locking an asset, we would see a confirmation of that locking, as shown in Figure 11-22.

■ **Note** The default contract loaded on Ivy is also the simplest contract. Your public key is used to lock the contract, and your signature is required to unlock the asset. It contains one clause called spend that verifies the public key and signature of the user to unlock assets.

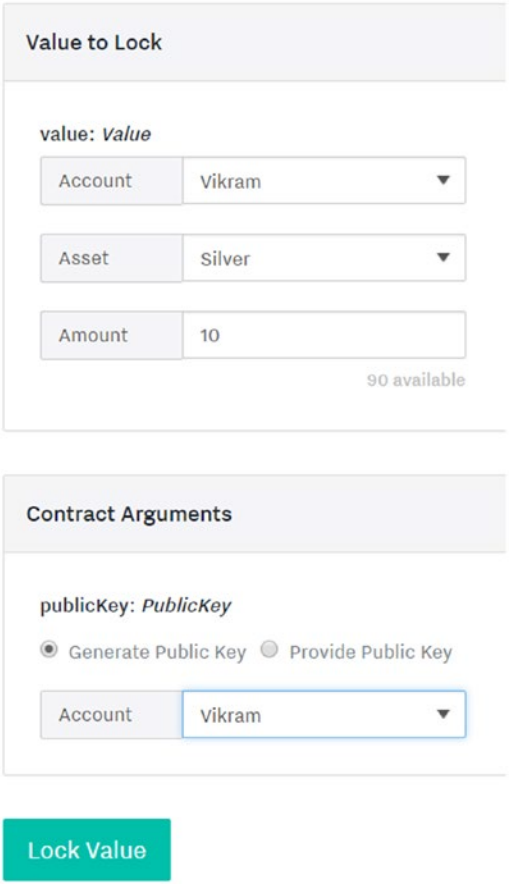


Figure 11-21. *Selecting asset parameters and contract arguments to lock an asset on the blockchain*

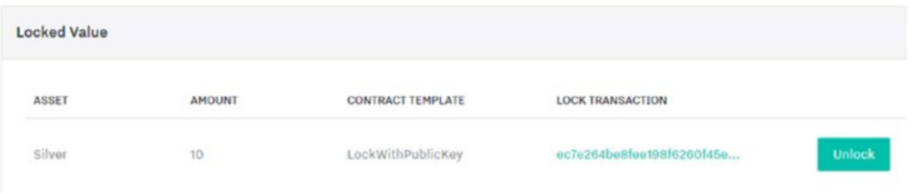


Figure 11-22. *Confirmation of asset locking using a key pair on the blockchain*

We presented how locking works in Ivy, now let's take a look at how to unlock this asset. Recall that we need a public key signature to unlock it and the key must match the account that locked it. Then, we can provide an unlock destination, as shown in Figure 11-23.

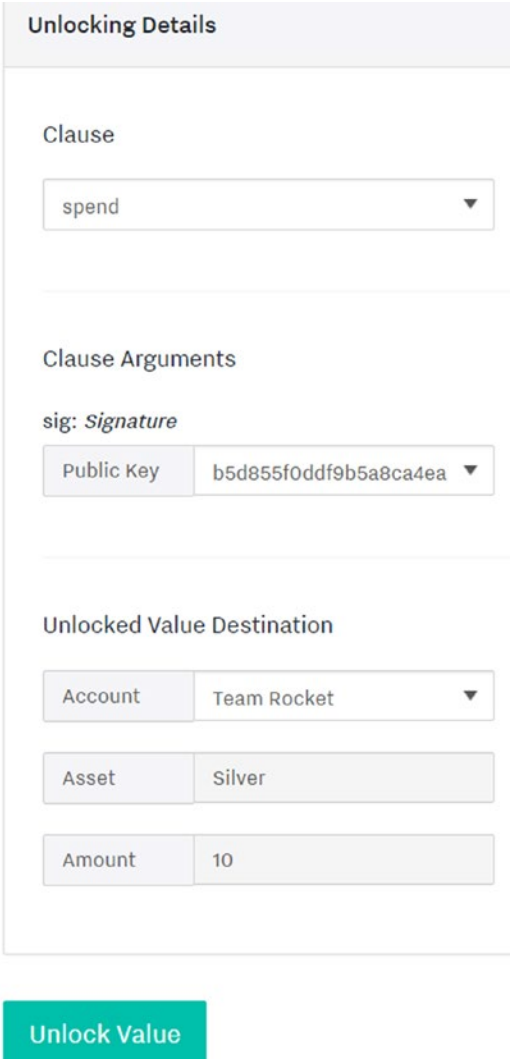


Figure 11-23. *Unlocking an asset using Ivy Playground*

Here, the unlocking is being done in conjunction with the spend clause to transfer the asset. We first have to provide the signature as an argument and then the destination for the unlocked value. This destination will control the amount of assets unlocked. Notice that this process is very similar to the trade we performed earlier.

This concludes our walkthrough of Chain Core and Ivy. We focused on how to create an HSM key, how to create new accounts, and the different modes of transactions available in Chain. Finally, we talked about using Ivy to add clauses in smart contracts that allow more sophisticated management of assets on the blockchain. Let's continue our journey with the next topic, Quorum blockchain.

Ethereum Enterprise Alliance

After the recent DAO hack (covered in Chapter 9), corporations such as JPMorgan, Microsoft, and Accenture formed a consortium called the EEA to make an enterprise-class Ethereum blockchain capable of handling high-volume transactions and business applications. JPMorgan has contributed massive development resources, providing the EEA with Quorum, an enterprise-grade permissioned version of Ethereum.

■ **Note** What is a permissioned ledger? A permissioned blockchain restricts the parties that can be block producers or miners, ultimately limiting who can contribute to consensus of the global state. In the financial world, this kind of verification might become necessary to scale a product for the masses.

Quorum's dev page on GitHub lists the following features that are an improvement over Ethereum:

- *Privacy:* A key innovation to make Quorum suitable for enterprise work is the introduction of privacy through private transactions and private contracts. This is accomplished using Constellation, a peer-to-peer messaging daemon that can direct the flow of private information to the appropriate network participants.
- *Alternative consensus:* Quorum does not rely on PoW/PoS for the permissioned ledger; instead, there are two consensus mechanisms better suited for enterprise-grade blockchains:
  - *QuorumChain:* A new smart-contract-based model that relies on voting by nodes to achieve consensus.
  - *Raft-based Consensus:* A consensus model suitable for closed-membership consortium settings, with very fast block-generation times.
- *Node permissioning:* A feature in Quorum that only allows connections to and from nodes that hold the appropriate identification keys, and have been registered to participate in the permissioned network.
- *Higher performance:* Quorum has to offer significantly higher performance than Ethereum to be used in a high-volume environment such as the one required to power bank-level transaction volume.

Let's dive into the features. We begin by talking about the key concepts behind the two consensus mechanisms available in Quorum. Then, we cover what modifications have made nodes capable of handling private transactions. Finally, we discuss Constellation and how it works to facilitate transfer of data in the

permissioned network. A summary of Quorum's features and private transactions created by JPMorgan follows our technical discussion. Let's begin with the consensus mechanisms.

- *QuorumChain:* This is a majority-voting consensus mechanism where a smart contract dictates which parties (nodes) can vote toward accepting or rejecting a block. The participating nodes on Quorum can either be maker nodes that actually construct a block or voter nodes that vote on the validity of a block. QuorumChain uses a signature verification method from Ethereum such as `ecrecover` to validate the votes received from maker nodes and voter nodes. Once the votes from voter nodes have reached a threshold, the block is accepted into the network.
- *Raft-based consensus:* The fundamental unit of work in a Raft-based system is a log entry. A log is an ordered sequence of events and it is considered consistent if all members of the network agree on the entries and their order. In Raft, a node can either be a leader or a follower; all nodes start out as followers. Eventually, through a peer-election process, one node emerges as the leader. Compare that to Ethereum where any node can mine a new block, or become the leader for that particular round. In Raft, only the leader can "forge" a block in the true sense, but the leader does not need to present any PoW. Instead, the leader proposes a block that the followers vote on. Once it receives a quorum (majority vote) of votes, it is accepted as the next block to extend the blockchain. The followers also send an acknowledgment to the leader, and now the block is committed to the log entry. Once the leader has received the acknowledgments, it notifies every node that this new entry (block) has been committed to the log.

Now that we have talked about consensus, let's look at what makes nodes accept private transactions. In Ethereum, all transactions are public, and by default, nodes only accept transactions that are broadcast publicly. For Quorum to work, the following changes have to be made:

- PoW consensus algorithms have been replaced by QuorumChain, which is voting-based. The ultimate goal is to have multiple consensus algorithms running on the blockchain in a model called pluggable consensus.
- There is a modified connection layer where nodes connected to the permissioned ledgers are identified and registered.
- The state tree has been split into two trees: a public state tree and a private state tree.
- Blocks can be validated with new logic containing private transactions.
- Transactions can be created with some data replaced with encrypted hashes to preserve privacy where required.

What propagates private transactions and contracts through the network and how is the flow of private data handled? For starters, there is a new optional parameter in the ETM called `privateFor`, and it can take multiple addresses. Quorum treats the addresses from this parameter as private in the network. A new transaction type, `IsPrivate`, has been introduced to mark certain transactions as private. However, the main tool making the propagation of private transactions possible is Constellation: a peer-to-peer encrypted message exchange. Here are the basics of how Constellation works within Quorum to provide privacy: Before a private transaction is propagated to the Quorum network, the message (contained within the transaction) and headers are replaced by the hash of an encrypted payload received from Constellation. Some participants in the network have had their public key included in the payload by Constellation. When those users receive the payload, they can decrypt it using their own instance of Constellation. Every other participant will only see an encrypted hash, and skip the transaction. The participants who are involved will decrypt the payload and send it to an EVM for execution, updating their internal state as a result. There

are two components in Constellation that play an important role in maintaining privacy: the transaction manager and the enclave.

The transaction manager stores encrypted transaction data, facilitates the exchange of encrypted payloads between participants, and manages the propagation of private transactions if the satisfying conditions have been met. It also makes function calls to other modules in Quorum, especially the enclave for cryptographic functions. In many ways, the transaction manager behaves like a central hub controlling the flow of messages from one component of Quorum to another. Enclave is the cryptographic core of Quorum. The transaction manager itself does not have access to any sensitive information or the keys, and it delegates cryptographic tasks such as symmetric key generation and data encryption and decryption to the enclave. It also holds the private keys for accounts in the network. Both the transaction manager and enclave are isolated from other components to enhance security.

zk-SNARKs

Zero-knowledge proofs, or Zero Knowledge Succinct Noninteractive Arguments of Knowledge (zk-SNARKs) are a new technology native to ZCash in which privacy and anonymity are maintained throughout the transactions happening on the blockchain. Recently, the EEA announced that the team behind ZCash will be helping implement a zk-SNARK layer for Quorum to enhance privacy and security of transactions on the blockchain. How do-zero knowledge proofs work? zk-SNARK technology is usually employed in complex payout situations. Let's look at an example. Say that Alice receives money from a smart contract if either X, Y, or Z happens. Now any of those situations (X, Y, Z) might be sensitive to Alice's health, or perhaps a business decision that she does not want to reveal to the public. This is where zk-SNARK can shine: It can create a proof to the smart contract that one of the three conditions (X, Y, Z) occurred, but not reveal which exact condition occurred. More formally, Christian Lundkvist from ConsenSys described zero-knowledge proofs as follows:

> *The goal of zero-knowledge proofs is for a verifier to be able to convince herself that a prover possesses knowledge of a secret parameter, called a witness, satisfying some relation, without revealing the witness to the verifier or anyone else.*

In our previous example, the three choices for Alice constitute a witness, and she had to prove that one of those occurred to the smart contract, but did not need to reveal the specific circumstances. Paige Peterson from ZCash confirmed that zk-SNARK would be coming to Quorum as a settlement layer, providing enhanced privacy and security to transactions. The eventual goal (described at Consensus 2017) is to make zero-knowledge proofs blockchain agnostic, and implement a settlement layer as a service. The future of integration with zero-knowledge proofs can be a paradigm shift for permissioned ledgers to carry out enterprise-level transactions. With some lessons learned, and a better settlement layer, there are some incredible opportunities for applications in health care to protect patient data and maintain privacy.

■ **Note** At the time of writing, a very interesting new development in the zk-SNARKs world was announced. ZoKrates is a higher-level programming language that can be used to write zero knowledge proofs (ZKPs) that compile to Ethereum-compatible ZKPs and executed on-chain.

Review of Quorum

Figures 11-24 and 11-25 are slides taken from a presentation by JPMorgan (with permission) and provide a high-level summary of Quorum given to the Hyperledger Project by David Voell.

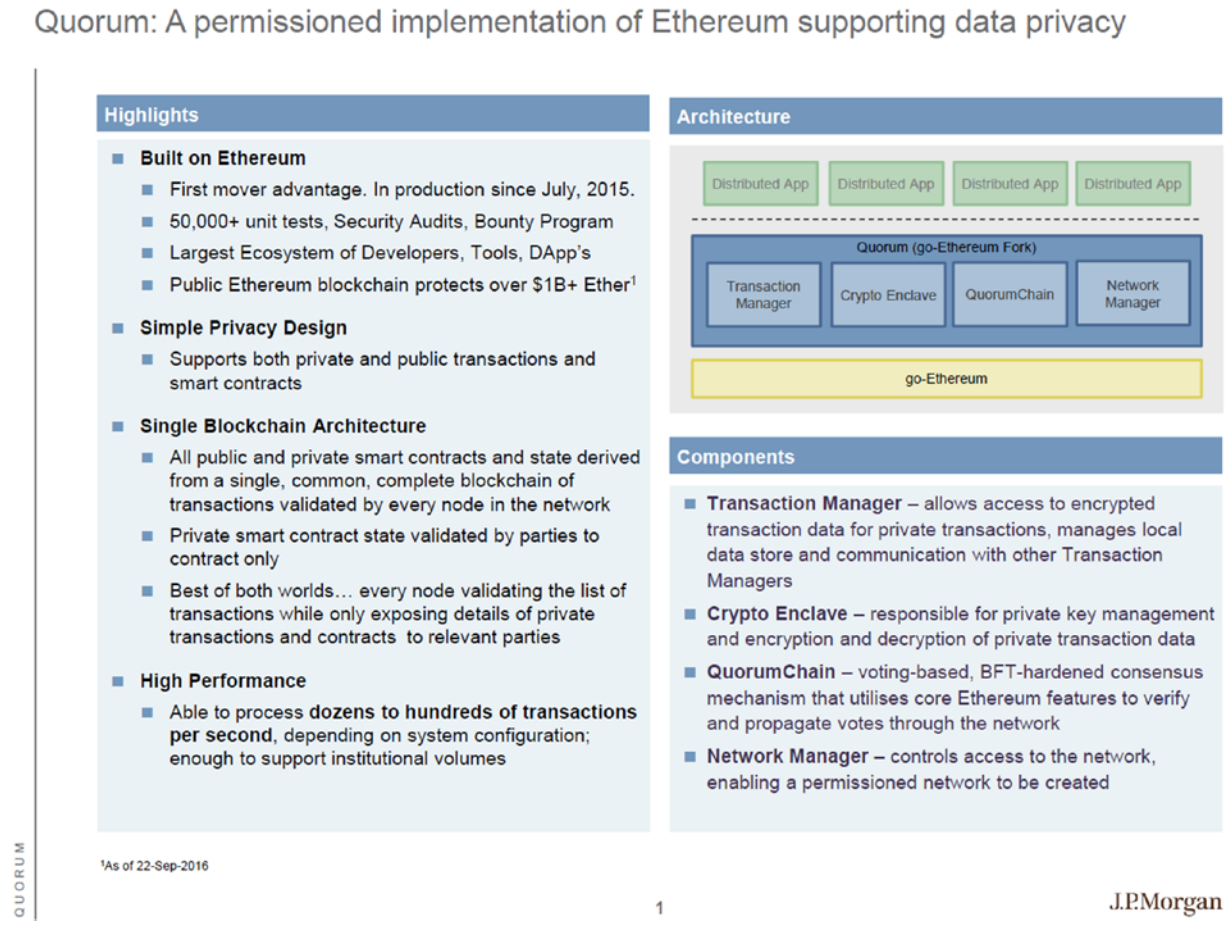

Quorum: A permissioned implementation of Ethereum supporting data privacy
Highlights
Built on Ethereum
First mover advantage. In production since July, 2015.
50,000+ unit tests, Security Audits, Bounty Program
Largest Ecosystem of Developers, Tools, DApp's
Public Ethereum blockchain protects over $1B+ Ether[1]
Simple Privacy Design
Supports both private and public transactions and smart contracts
Single Blockchain Architecture
All public and private smart contracts and state derived from a single, common, complete blockchain of transactions validated by every node in the network
Private smart contract state validated by parties to contract only
Best of both worlds… every node validating the list of transactions while only exposing details of private transactions and contracts to relevant parties
High Performance
Able to process dozens to hundreds of transactions per second, depending on system configuration; enough to support institutional volumes
Architecture
Distributed App
Distributed App
Distributed App
Distributed App
Quorum (go-Ethereum Fork)
Transaction Manager
Crypto Enclave
QuorumChain
Network Manager
go-Ethereum
Components
Transaction Manager – allows access to encrypted transaction data for private transactions, manages local data store and communication with other Transaction Managers
Crypto Enclave – responsible for private key management and encryption and decryption of private transaction data
QuorumChain – voting-based, BFT-hardened consensus mechanism that utilises core Ethereum features to verify and propagate votes through the network
Network Manager – controls access to the network, enabling a permissioned network to be created
[1]As of 22-Sep-2016
QUORUM
1
J.P.Morgan

Figure 11-24. *Overview of Quorum and its key innovations*

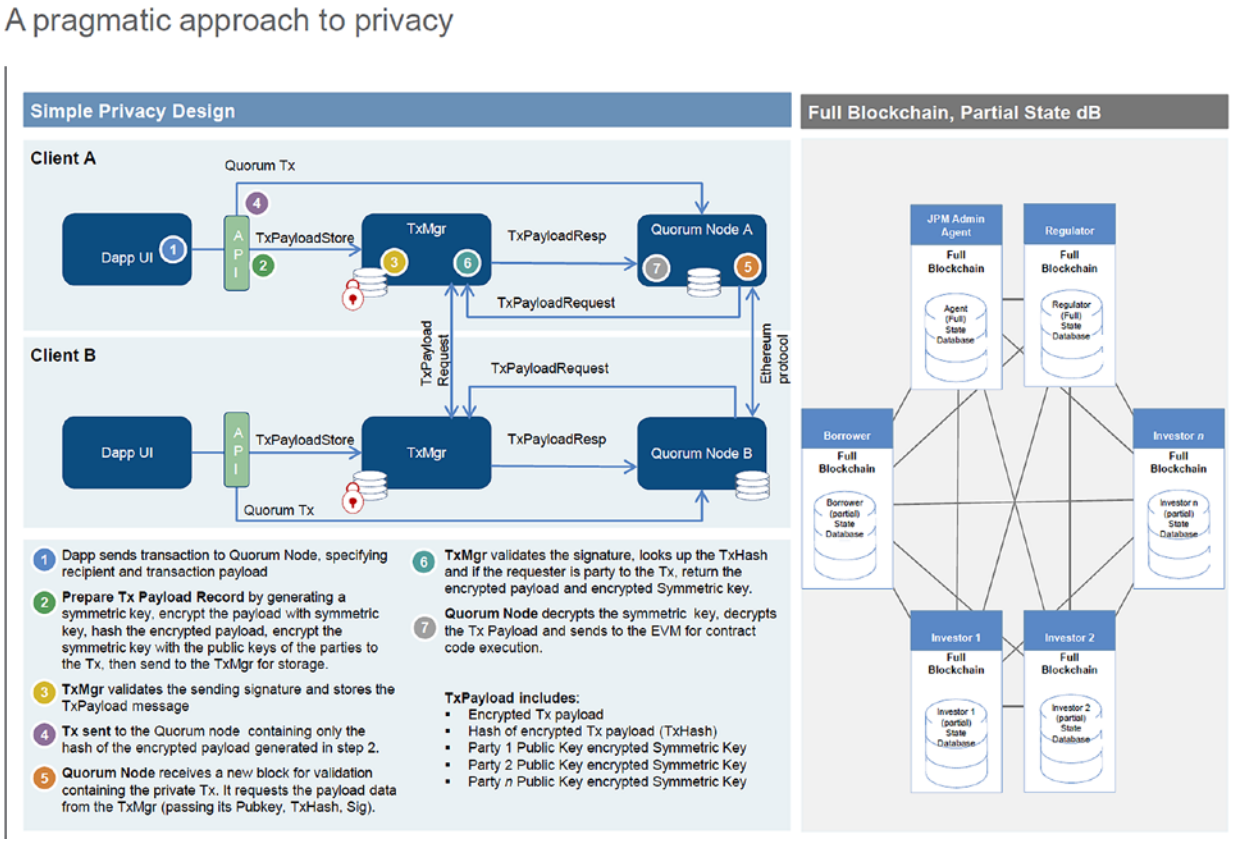

A pragmatic approach to privacy
Simple Privacy Design
Client A
Quorum Tx
Dapp UI
API
TxPayloadStore
TxMgr
TxPayloadResp
Quorum Node A
TxPayloadRequest
Client B
TxPayload Request
TxPayloadRequest
Ethereum protocol
Dapp UI
API
TxPayloadStore
TxMgr
TxPayloadResp
Quorum Node B
Quorum Tx
1 Dapp sends transaction to Quorum Node, specifying recipient and transaction payload
2 Prepare Tx Payload Record by generating a symmetric key, encrypt the payload with symmetric key, hash the encrypted payload, encrypt the symmetric key with the public keys of the parties to the Tx, then send to the TxMgr for storage.
3 TxMgr validates the sending signature and stores the TxPayload message
4 Tx sent to the Quorum node containing only the hash of the encrypted payload generated in step 2.
5 Quorum Node receives a new block for validation containing the private Tx. It requests the payload data from the TxMgr (passing its Pubkey, TxHash, Sig).
6 TxMgr validates the signature, looks up the TxHash and if the requester is party to the Tx, return the encrypted payload and encrypted Symmetric key.
7 Quorum Node decrypts the symmetric key, decrypts the Tx Payload and sends to the EVM for contract code execution.
TxPayload includes:
Encrypted Tx payload
Hash of encrypted Tx payload (TxHash)
Party 1 Public Key encrypted Symmetric Key
Party 2 Public Key encrypted Symmetric Key
Party n Public Key encrypted Symmetric Key
Full Blockchain, Partial State dB
JPM Admin Agent
Full Blockchain
Agent (Full) State Database
Regulator
Full Blockchain
Regulator (Full) State Database
Borrower
Full Blockchain
Borrower (partial) State Database
Investor n
Full Blockchain
Investor n (partial) State Database
Investor 1
Full Blockchain
Investor 1 (partial) State Database
Investor 2
Full Blockchain
Investor 2 (partial) State Database

Figure 11-25. *Overview of privacy and private transactions propagating the Quorum network*

Note the architectural design of Quorum in the top-right corner of Figure 11-24, just as we described all the major components earlier.

The most important take-home point from these slides is the detailed workflow of a private transaction as it propagates through the Quorum network. We did not have an opportunity to illustrate this workflow, so including the slide should enhance your understanding of how private transactions and encrypted payloads work in Quorum.

Ethereum Enterprise Roadmap

Figures 11-26 and 11-27 are taken from an update to the EEA provided by Bob Summerwill.

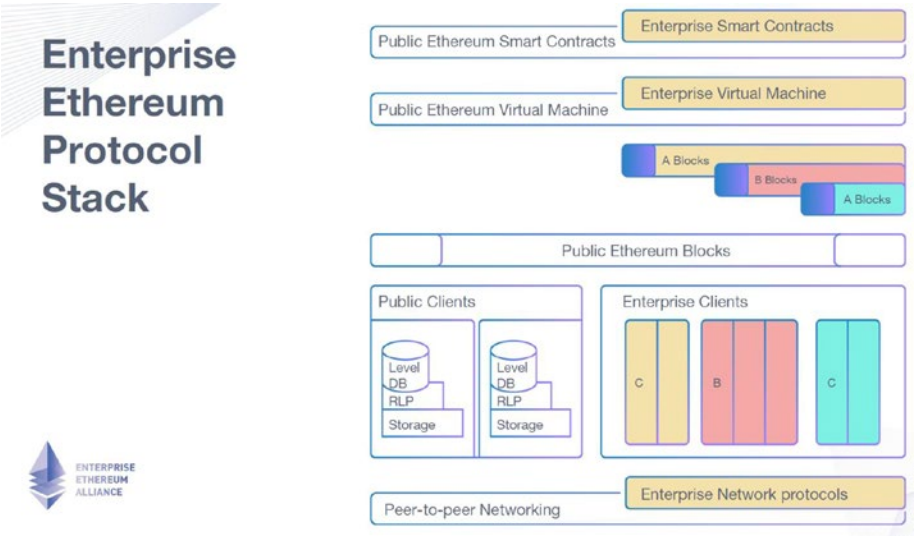


Figure 11-26. *A new protocol stack for Ethereum modified to handle enterprise-grade transactions*

In this example, privacy is important, and most miners (nodes) are known and trusted. Notice the enterprise protocols sit on top of a shared public–private blockchain with necessary modifications to preserve privacy. The public clients share the same storage, and virtual machines, whereas private or enterprise clients would be given isolated storage and virtual machines that execute encrypted payloads of instructions. More broadly speaking, Ethereum Enterprise is a bigger movement than simply Quorum being developed by the EEA. Let's take a step back and talk about why an enterprise version was needed. Jeremy Miller from ConsenSys provided three main reasons:

- Ethereum was developed initially for public chain deployment, where trustless transaction requirements outweigh absolute performance. The current public chain consensus algorithms (notably PoW) are overkill for networks with trusted actors and high throughput requirements.
- Public chains by definition have limited (at least initially) privacy and permissioning requirements. Although Ethereum does enable permissioning to be implemented within the smart contract and network layers, it is not readily compatible out of the box with traditional enterprise security and identity architectures or data privacy requirements.

- Naturally, the current Ethereum improvement process (dominated by Ethereum improvement proposals) is largely dominated by public chain matters, and it has been previously challenging for enterprise IT requirements to be clarified and prioritized within it.

What are the main technical objectives for EEA in 2017? Figure 11-27 provides three main focal points.

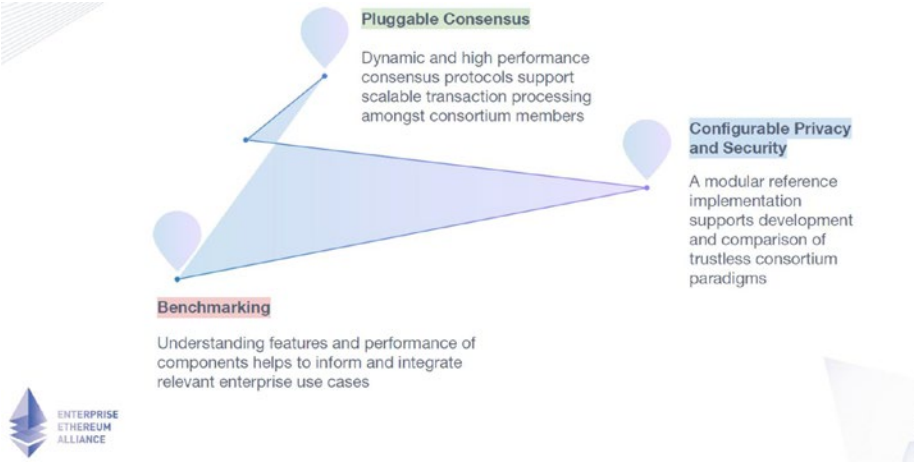

Pluggable Consensus
Dynamic and high performance consensus protocols support scalable transaction processing amongst consortium members
Configurable Privacy and Security
A modular reference implementation supports development and comparison of trustless consortium paradigms
Benchmarking
Understanding features and performance of components helps to inform and integrate relevant enterprise use cases
ENTERPRISE ETHEREUM ALLIANCE

Figure 11-27. *Three areas of focus for Quorum in 2017*

The largest focus and shift is toward providing a high level of privacy (identified by JPMorgan as a current major roadblock) that can be configured for transactions and smart contracts executed by users. A hybrid public–private chain can become the perfect model for consortium-type settings. The next focus is on creating a pluggable consensus model where transactions and complex situations can choose the best consensus mechanism applicable to the use case. Finally, high-volume performance is crucial for Quorum to succeed, therefore developing better benchmarking tools to measure and improve will become the key to wide adoption.

Summary

The rapid growth of research and development in the blockchain world can be explained by gaining a deeper understanding of where the value is created and captured for a traditional Internet companies and a blockchain company. Joel Monegro and Naval Ravikant talked about the idea of fat protocols, where most of the innovation in the blockchain space will happen at the core technology level. Then a token layer can monetize the use of underlying architecture and provide access to the application layer.

References

The main references used to prepare this chapter were the EOS developer guide and documentation, the Chain developer guide and API docs, Quorum docs and Quorum presentation to Hyperledger, and finally the Ethereum Enterprise Alliance roadmap presented by Bob Summerwill.

CHAPTER 12

Technological Revolutions and Financial Capital

The global financial markets are undergoing a drastic change that makes it clear that without innovation most business and financial models could soon become obsolete. A recent overview of the global financial system described the current system as a:

> *system that moves trillions of dollars a day and serves billions of people, but the system is rife with problems, adding cost through fees and delays, creating friction through redundant and onerous paperwork, and opening up opportunities for fraud and crime. To the best of our knowledge, 45 percent of financial intermediaries, such as payment networks, stock exchanges, and money transfer services, suffer from economic crime every year; the number is 37 percent for the entire economy and only 20 percent and 27 percent for the professional services and technology sectors respectively. It's no small wonder that regulatory costs continue to climb and remain a top concern for bankers. This all adds cost, with consumers ultimately bearing the burden.*[1]

Dan Tapscott pointed out that our financial system is inefficient for a multitude of reasons. Three specific realities are:

> *First, because it's antiquated, a kludge of industrial technologies and paper-based processes dressed up in a digital wrapper. Second, because it's centralized, which makes it resistant to change and vulnerable to systems failures and attacks. Third, it's exclusionary, denying billions of people access to basic financial tools. Bankers have largely dodged the sort of creative destruction that, while messy, is critical to economic vitality and progress."*[2]

We begin this chapter with issues related to the global financial markets and why blockchain can be an innovative solution for efficiencies in the financial markets. Then we move to the topics of venture capital, ICOs, cryptocurrencies, tokens, and exchanges. We address the significant ICO market impact, state

[1] https://hbr.org/2017/03/how-blockchain-is-changing-finance
[2] http://hrb.org/2017/03/how-blockchain-is-changing-finance

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7_12

of regulation, Securities and Exchange Commission (SEC) involvement, unique technologies, business models, RegTech, and related issues. As we review these concepts and issues there are several questions and thoughts to hold in mind:

- How does crowdfunding scale blockchain applications?
- How can new companies and lending platforms be created?
- Can RegTech answer the need in the market for efficient compliance?
- What is the market impact of FinTech for banking and investment banking?
- What is the state of ICO fundraising and the ICO bubble?
- How can people of all levels of wealth participate in financial markets as technology helps democratize financial opportunities?

We close the chapter with the state of multiple large financial groups and their state of involvement in blockchain, FinTech, and other financial technologies.

State of the Blockchain Industry

The state of the blockchain industry indicates massive growth in the second quarter of 2017. According to Smith & Crown, this proved to be a period of growth across the Blockchain industry. The cryptotokens markets rose, doubling and tripling in value in the span of a few weeks. The Smith & Crown Index (SCI) over this quarter reflected a bull market, more than doubling in value between April and June. The growth of capitalization in cryptotoken markets was accompanied by a frenzy of activity in the token sale market. By all accounts, the second quarter of 2017 has been a record-setting period.[3]

Blockchain Solution

As we have stated in previous chapters, blockchain is an innovative solution for disrupting the inefficiencies in the financial system. Kastelein noted that there are five basic principles underlying the blockchain technology that allows blockchain to change how financial market transactions are created. It's worth repeating here the five basic principles underlying the technology:

- *Distributed database:* Each party on a blockchain has access to the entire database and its complete history. No single party controls the data or the information. Every party can verify the records of its transaction partners directly, without an intermediary.
- *Peer-to-peer transmission:* Communication occurs directly between peers instead of through a central node. Each node stores and forwards information to all other nodes.
- *Transparency with pseudonymity:* Every transaction and its associated value are visible to anyone with access to the system. Each node, or user, on a blockchain has a unique 30-plus-alphanumeric address that identifies it. Users can choose to remain anonymous or provide proof of their identity to others. Transactions occur between blockchain addresses.

[3] https://www.smithandcrown.com/categories/feature/

- *Irreversibility of records:* Once a transaction is entered in the database and the accounts are updated, the records cannot be altered, because they're linked to every transaction record that came before them (hence the term *chain*). Various computational algorithms and approaches are deployed to ensure that the recording on the database is permanent, chronologically ordered, and available to all others on the network.
- *Computational logic:* The digital nature of the ledger means that blockchain transactions can be tied to computational logic and in essence programmed. Users can therefore set up algorithms and rules that automatically trigger transactions between nodes.[4]

Tapscott stated:

> *For the first time in human history, two or more parties, be they businesses or individuals who may not even know each other, can forge agreements, make transactions, and build value without relying on intermediaries (such as banks, rating agencies, and government bodies such as the US Department of State) to verify their identities, establish trust or perform the critical business logic—contracting, clearing, settling, and record-keeping tasks, that are foundational to all forms of commerce.*[5]

Blockchain applications can reduce transaction costs for all participants in an economy via peer-to-peer transactions and collaboration. Blockchain is truly a game-changing financial markets solution using new technology and empowered by thought leadership.

Venture Capital and ICOs

The real question is this: Will ICOs supplant traditional venture capital as a fundraising model? Few could ever imagine that in one year the venture capital industry could be outpaced and changed by new innovative methods of fundraising called ICOs. ICOs, also known as token sales, came together as blockchain technology, crowdfunding, innovative wealth ideas, and cryptocurrencies investing developed new models. ICOs are both a threat and an opportunity for the venture capital industry.

The traditional venture capitalist sees opportunities in ICOs for profits from cryptocurrency, blockchain investments, liquidity, and potential for faster financial gains. There might be a disruption in the way traditional venture capitalists operate and their position in the market. This is a great concern and time of change in the financial markets driven by technological innovation. The SEC recently asserted in a letter that ICOs are subject to security laws. The certainty that the SEC will act on cryptocurrency issues and ICOs clears up one big question. However, it is unclear how individuals, groups, and offerings become SEC compliant. The process will take time to resolve and for rules and precedents to be established.

Initial Coin Offerings

An ICO is a means of crowdfunding the release of a new cryptocurrency. Generally, tokens for the new cryptocurrency are sold to raise money for technical development before the cryptocurrency is released. Unlike an initial public offering (IPO), acquisition of tokens does not grant ownership in the company developing the new cryptocurrency. Unlike an IPO, there is no (comprehensive) government regulation of an ICO.[6]

---

[4] https://hrb.org/2017/03/what-initial-coin-offerings-are-and-why-vc-firms-care
[5] https://hrb.org/2017/03/how-blockchain-is-changing-finance
[6] https://en.wikipedia.org/w/index.php?title=Initial_coin_offering&oldid=784220634

ICOs and new funding models using distributed ledger methodologies are starting to disrupt both public markets (IPOs) and private investments (venture capital). An article from Coin Desk illustrated the blockchain's impact on venture capital formation. Coin Desk further noted, "There is proven demand and interest from both the entrepreneurial and investor audiences and limited regulatory guidance. ICOs could continue to gain steam as a funding mechanism."[7]

"Initial Coin Offerings (ICOs) are changing the cryptocurrency markets in rapid and expanding ways. Additionally, the venture capital industry is trying to understand this new financial investment. The bitcoin community created the situation by a convergence of blockchain technology, new wealth, clever entrepreneurs, and crypto-investors who are backed by blockchain-fueled ideas," stated *Harvard Business Review* writer Richard Kastelein.[8]

ICOs are dominating the overall crowdfunding charts in terms of funds raised, with half of the top 20 raises coming from the crypto community. The companies, such as Goldman Sachs, Nasdaq, and Intercontinental Exchange, the U.S. holding company that owns the New York Stock Exchange, which dominate the IPO and listing business, have been among the largest investors in blockchain ventures.[9]

An ICO is explained by Coin Telegraph as

> *a recently emerged concept of crowdfunding projects in the cryptocurrency and bBlockchain industries. When a company releases its own cryptocurrency with a purpose of funding, it releases a certain number of crypto-tokens and then sells those tokens to its intended audience, most commonly in exchange for Bitcoins, but it can be fiat money as well. As a result the company gets the capital to fund the product development and the audience members get the crypto tokens share, plus, they have complete ownership of these shares.*[10]

A specific example of how an ICO is created and the steps for execution are described in the SONM execution model. First of all, what is SONM? "SONM is a global operating system that is also a decentralized worldwide fog computer. It has the potential to include unlimited computing power (IoE, IoT). Worldwide computations organized using the SONM system can serve to complete numerous tasks from CGI rendering to scientific computations."[11]

The defining feature of SONM is its decentralized open structure, where buyers and workers can interact with no middleman, while building a market profitable for them first, unlike other cloud services (e.g., Amazon, Microsoft, Google).

Unlike widespread centralized cloud services, the SONM project implements a fog computing structure, a decentralized pool of devices, all of which are connected to the Internet (IoT/IoE).

SONM is implemented using the SOSNA architecture for fog computing.

The specific execution steps in the SONM ICO are as follows:

- The SONM platform uses a token of the same name, SONM (ticker SNM).
- Total supply of SNM will be limited to the amount of tokens created during the crowdfunding period.

[7] https://www.coindesk.com/ico-investments-pass-vc-funding-in-blockchain-market-first/
[8] http://hrb.org/2017/03/what-initial-coin-offerings-are-and-why-vc-firms-care
[9] http://hrb.org/2017/03/what-initial-coin-offerings-are-and-why-vc-firms-care
[10] https://cointelegraph.com/explained/ico-explained
[11] https://sonm.io/Somn-BusinessOverview.pdf

- SNM tokens will be used by the computing power buyers to pay for the calculations using the smart-contracts-based system.
- SNM is a token issued on Ethereum blockchain using standards of implementation and storage and management of tokens including Ethereum wallet.
- SONM project crowdfunding, ICO, and SNM creation will take place using Ethereum smart contracts.
- Participants willing to support the SONM project development will send Ether to a specified ICO Ethereum address creating SNM tokens by this transaction at the specified SNM/ETH exchange rate.
- ICO participants will be able to send Ether to the SONM crowdfunding Ethereum address only after the start of the crowdfunding period (specified as the Ethereum block number).
- Crowdfunding will finish when the specified end block is created or when the ICO cap is reached.
- SNM tokens sale ICO.
- SONM presale launched on April 15, 2017, and was successfully finished in less than 12, hours raising 10,000 Ethereum.
- Pre-ICO tokens will be transferred into the main token contract through a special safe migration function.
- Token allocations are completed to all parties.
- Transaction is complete.[12]

[12] https://sonm.io/Somn-BusinessOverview.pdf

This process is shown in Figure 12-1.

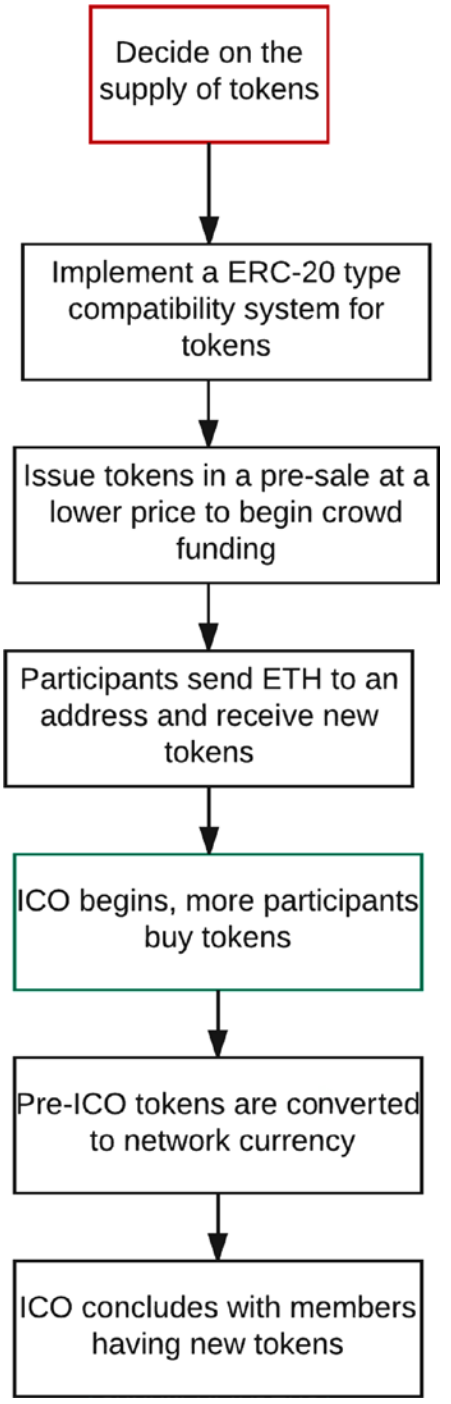

Decide on the supply of tokens
Implement a ERC-20 type compatibility system for tokens
Issue tokens in a pre-sale at a lower price to begin crowd funding
Participants send ETH to an address and receive new tokens
ICO begins, more participants buy tokens
Pre-ICO tokens are converted to network currency
ICO concludes with members having new tokens

Figure 12-1. *ICO process for the SONM token illustrated visually*

Digital Currency Exchanges

Digital currency exchanges (DCEs) or Bitcoin exchange are businesses that allow customers to trade digital currencies for other assets, such as conventional fiat money, or different digital currencies. They can be market measures that typically take the bid–ask spreads as transaction commissions for their services or simply charge fees as a matching platform.[13]

Typically DCEs operate outside of Western countries, avoiding regulatory oversight and complicating prosecutions. A U.S.-based global Bitcoin exchange named Kracken is located in San Francisco. The Kracken web site says Kracken is the world's largest global Bitcoin exchange in Euro volume and liquidity. Poloniex is described on its web site as a U.S.-based digital asset exchange offering maximum security and advanced trading features.

As one works with DCEs and for all aspects of operating in the world, it is very important to establish identity easily and exactly. The future of identity management looks different with blockchain technology in a decentralized digital world. Digital identity networks built on blockchain drives trust among businesses as a social enterprise by leveraging shared ledgers, smart contracts, and governance to standardized management, at the same time reducing the cost, risk, time, and complexity of decentralized identity management.

Status of ICO Regulation

At this time, Coin Desk discusses the state of ICO regulation with a focus on legal status in six nations. Several points of interest are quoted by a recent report from Autonomous Next, a FinTech and research firm. The report considers Switzerland and Singapore to be the two most advanced nations for creating welcoming environments for FinTech and cryptocurrencies. In Switzerland, the law is that cryptocurrencies are assets rather than securities. The same is true for Singapore. The Singapore MAS authority does not regulate virtual currency transactions, but does monitor KYC and AML, the report states.

The report singled out the United Kingdom and United States as jurisdictions with high activity, but a lack of legal clarity. The United States, with many regulatory groups and 50 states that implement rules, makes the process of regulation more complex. The state of Delaware has recently passed blockchain-related legislation. In China, tokens are considered a nonmonetary digital asset. Russia has been welcoming of cryptocurrencies. Cryptotokens are categorized as legal financial instruments similar to derivatives.[14]

Smith & Crown stated a significant problem in the legal status of token sales still being in question. Participants in token sales might not enjoy the same legal status or protections as investors in private and public equity sales. Uncapped token sale raises and structures that allow projects to raise large sums of capital while retaining majority control over their token economy could exacerbate this problem by drawing attention from regulatory agencies. A number of countries are actively exploring new regulatory frameworks for token sales, and several groups, including Smith & Crown, are developing guidelines for best practices and self-regulation.[15]

The current regulation status is that the SEC in the United States issued a letter stating that ICOs, cryptocurrencies, and related matters are viewed by the SEC as securities. As stated previously, this brings some clarity to the marketplace on this issue. Although some will read it positively and others negatively, ultimately the SEC statement set in motion an entirely new understanding of the regulatory environment both in the United States and globally.

[13] https://en.wikipedia.org/wiki/Digital_currency_exchange
[14] https:www.coindesk.com/state-ico-regulation-new-report-outlines-legal-status-6-nations/
[15] https://www.smithandcrowm.com/quarter-two-review

A Securities Law Framework for Blockchain Tokens describes multiple key thoughts and actions for anyone that is interested in blockchain tokens. "The Framework focuses on US federal securities laws because these laws pose the biggest risk for crowd sales of blockchain tokens. In many jurisdictions, there may also be issues under anti-money laundering laws and general consumer protection laws, as well as, specific laws depending on what the token actually does." The Howey Test establishes the test for whether an investment contract is a security (*SEC v. Howey*).[16]

The framework illustrates six best practices for token sales:

1. Publish a detailed whitepaper.
2. For a presale, commit to a development roadmap.
3. Use an open, public blockchain and publish all codes.
4. Use clear, logical, and fair pricing in the token sale.
5. Determine the percentage of tokens set aside for the development team.
6. Avoid marketing the token as an investment.

Pros and Cons of ICO Investments

The pros and cons of investing in ICOs are illustrated by Jim Reynolds in a recent article from "Invest It In: Investment Ideas." This list of pros and cons is by no means exhaustive, but it does contain many points to consider.

The following are the pros of ICOs.

For ICO Founders: Entrepreneurs

- Raise capital efficiently.
- ICOs are much cheaper than IPOs.
- ICOs require much less documentation than IPOs.
- Branding and marketing opportunity to get exposure for an altcoin.
- Community building.
- Create skin in the game with early adopters; this will make them part of the marketing mechanism of the project.
- Entrepreneurs share both the risks and the benefits of their efforts with the investors.
- Founders/developers have a method that can help them finance a project that can make the best use of their skills to the maximum possible extent.
- Well-respected crypto-experts have a channel to cash in on the skills and credibility they built over the years.
- Proof of stake altcoins resolve the problem of fair distribution through ICOs, and in PoS the coins come to fruition immediately.

[16] https://www.coinbase.com/legal/securities-law-framework.pdf

- Venture capital funding is much more intrusive on the founder's Vision360. An alternative to an ICO is borrowing, but this has many implications on the project cash flow that is not always possible to manage in altcoin/crypto projects.
- Some transparency; for example, an escrow can be used to verify how the funds are being spent after the ICO.
- Early investors will have more liquidity in early-stage companies.
- Early access to a token that has the potential for capital growth.
- Not regulated or registered with any government organization and there are usually no investor protections other than what is built into the platform itself.
- Investors can be part of a community.
- An innovative way to deploy capital.
- An ICO that uses existing networks such as Stratis, Ardor, and Ethereum, is tapping into the network capital of an existing ecosystem.
- Divest from main cryptocurrencies into altcoins.
- Investors are usually the first users of the altcoin; thus, unlike holding a stock of a company whose products an investor never used, ironically altcoins can be more tangible than other investments.
- The returns from investing in ICOs can be up to 1,000 percent, and also could be complete losses.
- Diversification into other assets.
- A high-risk, high-reward investment which is (to some extent) disconnected from the stock market and the economy.
- Own an alternative asset not based on fiat currency.

For the Cryptocurrency Community

- Altcoins will fuel the race to build Web 3.0, a decentralized web. The Internet stack becomes fully independent from any central entity.
- Altcoins are the cutting edge of FinTech, even if an altcoin project technology fails. There will be lessons learned regarding the technology and business model being proposed that will benefit the whole community.
- More competition in the crypto space makes the competitors leaner, meaning the "invisible hand" of the market frets much faster in terms of creative destruction and survival of the fittest the more altcoin projects are launched.
- Intra-altcoin competition is healthy, as it prepares the altcoins for the real competition, the crypto-based decentralized projects vs. traditional firms.
- There are two schools of thought: Bitcoin maximalists who consider Bitcoin as the one and only true cryptocurrency and altcoins as experiments. Others think the altcoins will eventually replace Bitcoin, just like the video was replaced by the CD, Myspace by Facebook, and old cameras by digital ones.

The following are the risks of investing in ICOs.

- Scammers take advantage of an unregulated industry.
- Amateurs can use ICOs to launch projects that are doomed to fail.
- Long project delivery timelines increase the risk that competitive products will be launched before.
- Exchanges need to accept the altcoin for there to be a market for the altcoin.
- An ICO can be surrounded by hype, and "pumping" to mask the cons of the ICOs and make investors invest emotionally, only to find out later that it was all hot air. This is done by piggybacking on the success of Bitcoin, Ethereum, and Dash without offering anything real back in return.
- Regulators can change rules and make coins with certain functionality illegal in the future.
- Altcoin technology is extremely new, and basic issues such as agreement on protocols are not yet established. Many altcoins will be born, and others will dissipate in the past until we eventually see the Google, Facebook and YouTube of cryptocoins.
- Certain tokens can be copied (forked) and made better. The clone can eventually have more value than the original. This happens when the token is not an intrinsic part of the network.[17]

Regulation Technology: RegChain

Regulation technology (RegTech) is a new innovation area that holds great promise for the regulation industry. RegTech is defined by Investopedia as "a blend word of regulating technology" that was created to address regulatory challenges in the financial services sector through innovative technologies. RegTech consists of a group of companies that use technology to help businesses comply with regulations efficiently and inexpensively.[18]

A summary in EY's publication "Innovating with Reg Tech" illustrates several benefits offering regulation technology:

- Supports innovation.
- Provides analytics.
- Reduces cost of compliance.

There are also some short-term benefits:

- Cost reduction.
- Sustainable and scalable solutions.
- Advanced data analytics.
- Control and risk platforms to be linked seamlessly.

[17] https://www.investitin.com/crypto-ico-pros-cons/
[18] www.investopedia.com/terms/r/regtech.asp

Long-term benefits include the following:

- Positive customer experience.
- Increased market stability.
- Improved governance.
- Enhanced regulatory reporting.[19]

A current innovation experience of a RegTech model created by Deloitte is RegChain.

> *Deloitte, in collaboration with Irish Funds and their members, advanced "Project Lighthouse" to assess blockchain technology's ability to service regulatory reporting requirements. The project tested the ability for a platform to provide individual nodes for fund administrators to store and analyze fund data while coding regulatory reporting requirements into smart contracts for execution and data validation. A regulator node was also facilitated, allowing the safe and secure exchange of data between firms and the regulator, as well as to increase overall reporting efficiency and market transparency. In addition to technical design and development, a comparative business analysis was undertaken to review the cost–benefit analysis of the proposed blockchain solution.*
>
> *RegChain was developed using Deloitte's rapid prototyping process, which uses an experiment-driven agile methodology. Key phrases included solution visioning definition of design and test parameters, development sprints, and ongoing reviews with an industry subcommittee with participants from across the fund administrator and fund management world.*
>
> *A key consideration and cornerstone for this project was to ensure collaboration among technologists and industry representatives from operations, regulatory teams, and senior management. This was deemed critical in order to have a comprehensive PoC design, and moreover, to help define how a future production solution could be realized.*[20]

Blockchain technology was used due to a number of features and characteristics that can enhance the overall ability to meet reporting requirements:

- Data integrity.
- Reliability.
- Storage and speed.
- Analytics.
- Proof-of-concept (PoC).

The PoC created RegChain, a blockchain-based platform that streamlined the traditional regulatory reporting processes by acting as a central repository for the safe storage and review of a large volume of regulatory data. RegChain has been used in the marketplace successfully in multiple applications and gives hope that the future will see wider adoption and enjoy the benefits of this model.

---

[19] https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/financial-services/performancemagazine/articles/lu_RegChain%20Reaction_Performance23.pdf

[20] https://www2.deloitte.com/content/dam/Deloitte/lu/Documents/financial-services/performancemagazine/articles/lu_RegChain%20Reaction_Performance23.pdf

New Blockchain Companies and Ideas

A *Harvard Business Review* article states, "Many businesses have yet to make the leap from the Industrial Age to the Information Age, and the gap between technological and organizational progress is widening."[21] Goldman Sachs has taken a series of steps that are bold and decisive in the area of digital innovation and blockchain applications. Goldman has been involved in some blockchain technology-based companies like Circle and Digital Assets Holdings. Additionally, in October 2016 Goldman Sachs introduced an online platform offering unsecured personal loans to consumers.

Homechain and SALT

Another new company is Homechain, described as the future of loan origination and regulatory compliance. The business idea is reducing the home loan origination and regulatory compliance process from 42 days to 5 days. RegChain allows for compliance to reporting bodies.

At this time, blockchain technology is at the stage where the Internet was in 1992, and it is opening up a wealth of new possibilities that have the promise to add value to numerous industries, including finance, health, education, music, art, government, and more.[22]

The SALT lending platform allows the holders of blockchain to leverage their holdings as collateral for cash loans. SALT is the first asset-based lending platform to give blockchain asset holders access to liquidity with them having to sell their tokens. SALT provides investors an innovative and secure opportunity to lend against a high-growth asset class through a fully collateralized debt vehicle. SALT is traditional lending secured by nontraditional collateral.

Each SALT token is representative of a membership to the SALT Lending Platform. The token is an ERC20 smart contract.

Blockchain technology is making waves in many industries. Companies in various sectors are innovating and using blockchain to create new applications and start up disruptive companies. A report by Accenture shows the cost data of eight of the world's largest investment banks, and states that blockchain technology could help reduce the costs of investment banks by as much as $12 billion per annum by 2025. There are multiple benefits for investment banks using blockchain technology, including safer data, more secure data, and cost reductions.[23]

Ambrosus, Numerai, and SWARM

Another example of a new company that was recently created is a Swiss blockchain company, Ambrosus. This company was launched to employ smart contracts to track food quality. Innovation using blockchain technology is being adapted everywhere.

Numerai is a new kind of hedge fund built by a network of data scientists. Numerai is a crowdsourced hedge fund for machine learning experts. In the first year of collection, 7,500 data scientists created algorithms on the Numerai platform as reported by Tech Crunch. Numerai announced Numeraire, a cryptocurrency token to incentivize data scientists around the world to contribute AI. The Numerai smart contract was deployed to Ethereum and more than 1.2 million tokens were sent to 19,000 data scientists around the world, as reported by Numerai.[24]

The startup company Swarm advanced several new concepts into the marketplace and was at the leading edge of two emerging concepts, crowdfunding and cryptocurrency, while creating a startup in a market with complex and evolving regulations. The idea was to transform the way entrepreneurs raise money. Swarm created a new idea called crypto-equity, a token that represents the success of your project.

---

[21] https://hbr.org/2017/03/how-blockchain-is-changing-finance
[22] https://hbr.org/2017/03/how-blockchain-is-changing-finance
[23] https://saltlending.zendesk.com/hc/en-us/sections/115002568808-Technology
[24] https://medium.com/numerai/an-ai-hedge-fund-goes-live-on-ethereum-a80470c6b681

Swarm was built on the following three components:

- Crypto-equity.
- Crowdsourced due diligence.
- Coins distributed to all Swarm holders. The best description of Swarm is like a crypto-Kickstarter offered via coins.

KICKCO is a new company with a unique model. Their web site states:

> *KICKCO sits at the intersection of two young industries: blockchain and crowdfunding. KICKCO moves crowdfunding from centralized platforms such as Kickstarter, to Ethereum-based smart contracts. This not only allows us to implement the crowdfunding model in a decentralized way—significantly reducing overhead—it also provides a mechanism to protect backers from failed projects—guaranteeing their investment with blockchain based tokens called KickCoins. KICKCO will provide users with a powerful, convenient, and up to date platform for both ICO and crowdfunding campaigns. KICKCO is a site for automated and independent ICOs, pre-ICOs, and crowdfunding campaigns built on Ethereum and funded by cryptocurrencies. The purpose of KICKCO is to solve the aforementioned problems and create a single platform that will unite the creators and backers of ICO and crowdfunding campaigns to form an active up to date community.*[25]

The companies that have been described show some of the innovation and creative thought that works around the blockchain models. The blockchain continues to advance new and exciting ways for financial markets to become more efficient.

Democratizing Investment Opportunities

Blockchain can help the world's poorest people. A World Economic Forum article described the way that transactions can be recorded in an inexpensive and transparent way, allowing for money to be exchanged without fear, fraud, or theft. Blockchain smart contracts, sending money internationally, reducing costs, insurance services, helping small businesses, humanitarian aid, and blockchain-powered identity systems are just some of the ways blockchain can help many people. For those without passports, birth certificates, phones, or e-mail, blockchain records can speed processing, allowing for a better way of life for many people.

Through crowdfunding smaller investment amounts and many new capital formation ventures, people who have previously not had the opportunity for sharing in the economies of a growing financial market can participate. Blockchain helps to democratize finance and affect lives in new ways.

[25] https://www.kickico.com/whitepaper

Summary

The future is for creation, innovation, change, and global impact as crowdfunding enables blockchain on a global scale. Reports from both EY and Innovate Finance delve into the capital markets landscape. Chris Skinner, in his blog, "Fintech and the World of Investment Banking," stated that, "there is a raft of new technologies that are architecting the landscape of capital markets and hundreds of start-ups leveraging these new technologies to both assist and attack the inefficiencies in the investment banking world. Rather than ignoring these changes the biggest banks are investing in them."[26]

Goldman Sachs, Citicorp, JPMorgan, Morgan Stanley, Wells Fargo, and Bank of America are just some of the large banks and investment bankers investing millions in FinTech, blockchain, and other technology innovations.

Disruption in the capital markets is moving at a rapid pace. A recent *Forbes* article indicated that there are more than 900 cryptocurrencies in existence now, with more being added each day. The ICO market has raised more money than the venture capital market in the last six months as of mid-2017. There will be many future market ups and downs, with new regulations coming to the market, and many changes on the way as we experience technological revelations and financial capital changing rapidly.

A special salute goes to the innovators, dreamers, individuals, and groups that are in the marketplace and see the future of a better financial world for everyone. The incredible commitment of incubators, accelerators, universities, and investment firms working toward blockchain efficiencies and innovation will change the way the world views financial capital.

Examples of innovation are everywhere. Goldman Sachs, JPMorgan, Nasdaq, and special alternative investment firms such as Triloma Securities offer unique products, services, innovation, and capital. What is occurring in many local markets is a collaboration of groups configured and aligned working together for common goals in the technology area. One unique group in the Orlando and central Florida area has come together to help startups and advance blockchain research and applications. This group is comprised of various organizations including Florida Angel Nexus, Merging Traffic, StartUp Nations Ventures, Institute of Simulation & Training, METIL (Mixed Emerging Technology Integration Lab), the Medical Tourism Association, and many others, all working in a robust public–private partnership ecosystem that drives change for the good of all in the marketplace.

Blockchain will bring a unique technological revolution in financial capital.

[26] https://www.thefinanser.com/2017/08/fintech-world-investment-banking-html/

APPENDIX A

Building a Health Care Consortium

Authors: John Bass, Corey Todaro

Editors: Vikram Dhillon, David Metcalf

John Bass is the founder and CEO of Hashed Health, a blockchain-based health IT consortium with the members working to commercialize appropriate use cases in health care and deliver proof-of-concept that can be scaled. In this Appendix, we reproduce a long-form Interview with John Bass about topics such as these:

- His background and previous work in health care.
- The makings of Hashed Health.
- Collaborative and consortium models.
- Working groups for high-risk, high-reward model.
- Governance in Hashed Health.
- Member participation.

Note John Bass contributed to this interview and chapter in his personal capacity. The views expressed are his own and do not necessarily represent the views of other authors, employers or any affiliated institutional entities.

Vikram Dhillon: So John, can you tell our readers about your background, and what made you interested in building health care startups?

John Bass: Many of us in the health care technology industry have spent our professional lives working to build applications and intelligence that helps stakeholders (providers, vendors, payers, consumers, etc.) understand the opportunities to improve care delivery and curb costs. As a community, we have a hope that we can change the system; that we can make an unfair system fair again; that, using technology, we can go back to a place where health care is centered on the individual consumer patient rather than the corporate interest. We all dream of the ever-elusive health care platform that captures the hearts and minds of patients, builds the network effects, and makes everything better. We dream of saving our country from runaway health costs, a national security issue of our own making. We dream about delivering care to the millions of people who struggle with access. We dream of creating something so viral that it scales globally and helps millions of people around the world get the care they need. We want it so bad we lose sleep. We want it so bad we abandon our families for weeks at a time, hoping to light the fuse.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017

V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7

Vikram: What got you started thinking about building a blockchain-based health care company?

John: The startups I have been a part of have all focused on Web-based products that promoted collaboration, data sharing, shared workflows, and cross-enterprise clinical or financial performance. Our teams have built B2B platforms, patient portals, collaborative workflow solutions, surgical performance solutions, and other trendy IT solutions that I thought would change the industry. These efforts, like most of the solutions you see at the big health care technology trade shows, focus on adding applications on top of the existing infrastructure. They essentially support the existing health care value chains, with the goal of making them more efficient. Although successful at solving problems, it's clear that these years of work and millions of dollars have little impact on the macro cost and quality issues facing our country. The killer app and the platform are still out there.

After 20 years I have realized that there is a more fundamental problem that can't be fixed through additive technologies. Step back and you see that, over time, we have built a system that is more and more dependent on middle men, relational databases, and their associated applications. The structural truth is clear to me now. The structural truth is the driver of unstable economic trends, unfair access, failures in preventable illness, suboptimal outcomes, rising out-of-pocket costs, and medical debt. All systems inherit the characteristics of their container, which in the United States is a fee-for-service marketplace that lacks any sort of rational, free-market characteristics. Anything built within this container is constrained by those characteristics. I don't believe we can really solve anything building within the existing container.

Vikram: I think we would both agree that the existing container has worked well for now, but it's not resilient enough to work for the problems that we will be facing in the near future. What are your thoughts on this?

John: Don't get me wrong … the U.S. health system has been remarkably successful. More people die of old age than infectious disease. We have become experts at treating acute-care injuries and illnesses. Our pharmaceuticals industry produced a vaccine to Zika in a remarkably short amount of time. I would not say our system is broken or in need of a do-over. It is working exactly as we designed it to work. It is a representation of the user stories and requirements that we drew up for its developers. People from around the world travel to the United States to receive world-class treatment. And we are rapidly exporting our (good and bad) expertise to the rest of the world.

What is clear is that we are no longer getting our money's worth. The United States is approaching $5 trillion in spending and one third is waste. Medical errors are the third leading cause of mortality behind heart disease and cancer. Significant populations of people in the United States and around the world have poor access to care. We all live in environments where unhealthy influences, such as the consumption of sugar, have been institutionalized. Yesterday's system we created was not designed to counterbalance the behaviors that have emerged as the leading causes of today's cost and quality concerns. We are realizing that can't be fixed with new applications on top of the as-designed system. We have to rebuild, taking advantage of our successes while optimizing for today's social and economic truths.

Vikram: So what does the rebuild look like? What are the new user stories and requirements? Who are the new developers of this system? Can the current corporate interests be trusted to redesign a platform for the future? How do we get from here to there?

John: Hashed Health sees the blockchain as the protocol for the next generation of health solutions. By moving trust to the protocol layer, we lay the foundation for leaner, more agile care delivery models, payment systems, value chains, and consumer experiences. It is not *the* answer to these questions. It allows innovators to have a new conversation free of today's limiting factors. When combined with IoT, machine learning, artificial intelligence, cryptocurrency, design thinking, and other tools, blockchain becomes a recipe for true disruption.

The path to real change in health care is through the consumer. One reason blockchain is an enabler of disruption in health care lies in its potential to change the consumer's relationship with existing systems of health and wellness. It provides an alternative to today's corporate-focused environment that has become a limiting factor for society. Users will interact with this new model through their mobile devices, which will leverage "wallet" software that will become more and more self-sovereign over time. Consumers will

become prosumers with controls over their health assets. Patients will no longer be asked to join a series of siloed patient portals as they move through the continuum of care. This provides a more holistic approach across an individual's related health, wellness, social, and socioeconomic events. We have learned that one's health is so much more than the 5 percent fragments that happen during a clinical visit. Patients will be empowered to control their information and have the ability to donate or monetize their records as they wish. Over time, honeypots of medical records data will be decentralized and distributed, making the user's information less susceptible to theft.

Vikram: By many standards, health care IT moves as a slow monolith. This might be due to design, but can you talk a bit more about why we don't have a straightforward consumer-to-provider relationship in health IT, and health care more broadly?

John: The consumer-facing side of health IT is complicated. Consumerization of health care has been a buzzword for years now but has proven elusive. There are a few important reasons why this is true. First, on many levels, health care is not like other markets. Health care is not a traditional commodity or service to be purchased. Even if they were completely free to do so (more on this later), consumers are not in a position to make the best judgments about purchasing health care, either because of the complexity of health care treatments or due to the emergent need to seek care. Additionally, unlike almost all other products or services, a consumer cannot always predict the results of consuming health care. For example, purchasing a car produces predictable results: owning transportation equipment with stated features and in regulatory compliance. By contrast, a given treatment might or might not adequately address the illness or condition.

Beyond these economic basics, consumerization is complicated by the configuration of the health care system itself. The drive to "Uber-ize" health care runs head-on into the wall of the U.S. payment and delivery systems. The consumer has no knowledge of price for services. Providers themselves often cannot give pricing in advance. Furthermore, the consumer is not a party to the pricing agreement, as the contractual relationship is between the provider and the payor, on behalf of the consumer. These are all well-known, almost hackneyed issues related to U.S. health care. By themselves, they do not prevent the creation of potentially useful consumer tools. Instead, to have utility, consumer tools need to "plug in" to the systems that deliver and pay for health care services. Given the dizzying array of providers, payors, and service providers, as well as their oft-conflicting motivations, consumer tools are often stymied by the unwillingness of enterprises that control the health care marketplace. Consumerization of health care is elusive, not because of a lack of tools, but rather due to the lack of an open platform.

As groups of consumers organize or are convened in traditional and innovative ways, these groups will represent new centers of gravity for the future marketplace. The community or group (however you define it: village, employer, values-based group, nursing home, etc.) will also be empowered through the ability to organize, aggregate, and share. Similarly, groups of providers on the sell-side of the marketplace will organize (or be convened) into either traditional or innovative new cohorts for the provision of care in ways that are better, faster, and more economically rational than what can be delivered under today's bloated market structure.

Vikram: How do you think the large enterprises (provider groups and insurance companies) will evolve to respond to the new opportunities such as blockchain?

John: Enterprises will, over time, be affected by new opportunities and challenges that result from changing market structures. The effects will be felt throughout today's health care value chains, such as supply chain, revenue cycle, claims life cycle, clinical research, insurance and benefits, and clinical episodes. Audit and compliance activities will fall away to automation. Contracting, rebating, administrative fees, and discounting schemes will disappear or be transformed through the use of immutable, transparent shared ledgers. Closed relational databases like EMRs, ERPs, and materials management systems will evolve as the nature of their accounting functions move to the market level. Clinical trials programs will be optimized through sharing of information and streamlined administrative processes. Security will be enhanced and, over time, data honeypots will fall away to a decentralized structure.

Companies will not throw away their existing databases (at least not at first). First, these legacy systems will transact with the blockchain through APIs and then, over time, we will begin to see new, nimble, ground-up systems emerge. These systems will transform how we currently think about the movement of health, data, clinical, and financial assets across a market.

Machines will also begin to play a larger role in health and wellness. We already see a role for connected health machines, wearables, and a host of Web-enabled monitoring devices. Blockchain provides the foundation for a more secure and scalable health IoT environment. Devices can be registered and validated more scalably using the blockchain as a shared ledger. As a result, information from these machines can be trusted. Transfer of ownership can also be tracked more easily and the operational life cycle of the device can be recorded. Perhaps most interestingly, the device itself can be given a wallet and a set of smart contracts that allow it to execute commands, transfer assets, and operate in ways that drive clinical and financial value for consumers or for the machines themselves!

This vision of the consumer, the community, the enterprise, and the machine is truly disruptive. It's also a vision with many technical and nontechnical challenges.

Vikram: How do we get to the future that you described? What is the path to the consumer-driven future? How do we create demonstrations that prove the value of the technology? How do we get people on board?

John: Health care is a fast-follower. It is historically resistant to platforms and "uberization" because of regulatory considerations, complexity, and aversion to risk. Blockchain technology is new. Satoshi Nakamoto released the Bitcoin whitepaper in 2009. Ethereum was created in the summer of 2015. Other protocols being considered for health care projects, such as Hyperledger Fabric, Tendermint, and other protocols, are even newer. EOS, Tezos, and NEO just arrived on the scene with their record-setting token sales and new ideas around governance, consensus, and scalability. Smart contracts are still not very smart and are even less secure. It's anyone's guess who the winners will be in five to ten years. If given the choice, the chefs in health care's institutional kitchen would prefer to sit back and watch the technology marinate for a while.

The real limiting factor may actually be the nontechnical concerns. First, for blockchain solutions to be effective, you need a collaborative network of participants. This is why you read so much about consortia and blockchain. Without the consortium or the minimally viable network of participants, enterprise blockchain efforts are merely expensive academic exercises. Even if you have the network required, successful governance can be tricky. These collaborative concepts require some getting used to for some, a paradigm shift for most. Second, for certain use cases there are serious regulatory concerns that complicate productive use. Many great ideas will be limited by a regulatory framework that never imagined the decentralized marketplace. Third, there is still a significant amount of education and organization of the market required. After a year of operating our consortium, the majority of the market still needs consulting on what blockchain is, how it could affect their market structure, and why networks are key to success. Fourth, we believe that the early efforts to establish blockchain and distributed ledger technologies as a viable solution will require simple demonstations of value. These early demonstrations will not be designed to change the world. They will solve simple, unsexy problems in today's health system. As a result, there will be early complaints that these products do not deliver the value that blockchain had originally promised.

It will take some time to sort things out. For those of us who are early in the health care blockchain space, these are the complicated realities we face. The real question is timing.

Vikram: So, about the timing, how long do you think it will take for blockchain to become mainstream? How can we accelerate the time it takes to cross the chasm?

John: I often tell people that starting Hashed Health is the most excited and the most scared I have been in my professional career. From an innovator's perspective, the blockchain health care space is a dream. There is no template. Everything is new. The technology is new and immature. The collaborative innovation business model is new. The concept of "fat protocols" is new. We have to not only build products, we also have to build a market. We are innovating from every angle. Every day is a 3D chess match. It's high-risk, high-reward research and development. For Hashed, it's the most exciting opportunity imaginable.

We chose the collaborative business model because it is in line with the spirit of the blockchain and it seemed like the only path to success when we started the company in 2016. Unlike previous startups, this was not a "build a product, sell a product" model. Although we had many strong use cases in the queue, we knew we could not place all our bets on one. The market wasn't ready. We knew we needed to do more with less. The collaborative model allows for us to join together with industry thought leaders who can contribute to our ecosystems of shared value. By being a part of several successful projects with the companies that are using the products, we increase our odds that our products will gain productive use. We also lower the risks and increase the rewards for our customers. It was the only way, especially in a new and complex market.

Once we made this decision, we were quickly branded a consortium. We have never been comfortable with this label, although we struggled to come up with anything better. We are more like a mesh network or a product studio. At the end of the day, Hashed Health is a products company with many successful solutions in our portfolio. In many ways, any company building blockchain products is a consortium if they want to be successful. The network is often more important than the product itself. It's the relationship between the two where we spend a lot of our energy, because that's where success lies.

Vikram: So how do we start building the network? Currently, there is a lack of awareness and understanding in the health care space about blockchain technology. How do we bring the providers up to speed and get them interested?

John: To begin, we knew we needed to do a lot of thought leadership work to educate and organize the industry. We researched the technology heavily and we connected what was happening outside health care to our professional and personal health care experience. We invested heavily in research. After a time, we began our thought leadership work. We blogged, wrote newsletters, and spoke at conferences. We listened to the feedback we received and we iterated on our ideas and early products. We used those early products to recruit the initial members. At first we developed really complicated contracts, and then we simplified them to make getting started as easy as possible. We evolved into a structure that can flex governance, business, and technical requirements as needed based on the workgroup and the product. This allows us to meet a customer where they are, rather than forcing them down a contractual pathway that might be scary or uncomfortable. We started with larger organizations and we are now opening membership and workgroup activities up to earlier stage companies, universities, entrepreneurs, and thought leaders so that we can bring more contributors into the emerging networks.

Vikram: Can you talk a bit more about how the Hashed Health model benefits the members? What's the current structure of Hashed Health?

John: Our goal with our members is to build community and products that make meaningful use of the strengths of blockchain and distributed ledger technologies. We seek out members who will contribute to a project and a conversation, not just look to extract value from the group. We consider ourselves a health care company first, so the projects we work on must have the patients' best interests in mind.

Our members usually engage as general members who need some consulting before they are ready to build. The level of business and technical expertise inside health care institutions is very weak, so a consultative approach is required. Members usually have a concept or two that they believe they'd like to tackle, but they always require some guidance and some preparation prior to development. By providing health care expertise and blockchain expertise, we are able to deliver a unique product that drives productive use case concepts forward in a more efficient manner. By creating collaborative agreements, we can build incentives that support accomplishing rational milestones on the pathway from concept to productive use.

Each of our networks has its own constituents, cadence, business plan, technical plan, governance structure, and personality. Each is focused on solving a specific business problem, and we choose the protocol that fits the business need. Currently we have built products and demos on Hyperledger Fabric, Ethereum, a commercial platform called BitSE, and Tendermint and Ethermint. We are currently researching other protocols and various middleware products including Gem, Bloq, Nuco, Stratumn, Tezos,

EOS, and IOTA. The workgroups all share common learning and best practices that translate across projects. We have experts (either in house or contracted) who can support any discussion (technical, business, legal, regulatory) required to advance a product.

To cross the chasm, we feel it is important to begin with simple demonstrations of value. The simpler the better. We prefer projects that are not highly political, that do not require protected information, and that solve a problem in today's environment while laying the foundation for our vision of the future.

Vikram: Let's get into the portfolio of operational working groups within Hashed Health and the current target areas. Can you elaborate on some of the use -cases for us and how they're using the blockchain?

John: At the time of writing, Hashed Health has five active enterprise groups.

1. Provider Identity
2. Patient Identity
3. Payments
4. Supply Chain IoT
5. Clinical IoT (Wearables)

We are also in the process of forming several new enterprise groups that we expect to have operational soon, including:

1. Disease Registries
2. Clinical Trials
3. Medical Records
4. Pharma Revenue Cycle
5. Enterprise Resource Management Systems

These are areas where the Hashed team has both expertise and customer interest. The general membership dues support the initial business case and technical research. Once there is a decision to build, customers are required to enter a secondary development agreement specific to the project.

Perhaps the most popular example of our work is decentralized physician identity. Provider identity and its related data is foundational to the delivery of care today and in the future. From graduate medical education to stat licensing, medical staff credentialing, and payer contracting, the ready availability and reliability of data about a provider's identity, credentials, and reputation are paramount to ensuring patient safety and high-quality care. Globally, the world is facing a shortage of qualified health workers. That shortage is estimated to be 7.9 million and it is expected to grow to 12.9 million health workers by 2035. A crucial challenge amidst this shortage is the ability to identify, locate, and communicate with workers in remote locations. The single provider's identity is a complex tangle of data points. There are multiple elements held by multiple disparate stakeholders, such as medical schools, state licensing boards, and more. Some elements remain static over time (e.g., graduate degree), and others are dynamic over time (e.g., licensure, affiliation, residence, and contact information).

In this use case, we treat data fields as individual data assets. Providers and credentialing stakeholders jointly manage a distributed provider profile registry. Cryptographic signatures ensure primary source verification of essential credentials and certifications, allowing distributed networks to share real-time updates of crucial data. This process significantly reduces time, expense, and wasteful manual processes that happen today. This use case is attractive because it is fairly straightforward technically, it is not political, the data are not sensitive, and no key stakeholder has a competitive interest. We would argue that it is a good blockchain use case because provider identity is not centralized and there are currently trust and incentive issues that need to be overcome. Although there are current efforts underway to centralize this information,

the various data elements are not centrally granted, administered, or consumed. An easily auditable market-level data structure will deliver trust and efficiency in a market that currently has neither.

In this example, you can see how Hashed Health has seeded a market with a simple product built on a simple, yet impactful use case. Blockchain allows us to tackle this problem in a way that was not previously imaginable.

We are equally excited about our other foundational use cases. We are working with providers on patient registration authority products. We are working with a multiconstituent workgroup on an exciting payments model that links payments to benefits and behavior. We are working with government institutions on public health surveillance and clinical trials decentralization. We have begun an exciting journey that continues to pick up momentum, new ideas, and additional expertise in both blockchain and specific health care subject matter.

Vikram: One major problem with blockchain integration into health care is concerns about privacy. Blockchain was designed to be anonymized, but for health data, we need this interesting combination of trusted miners, completely private transactions, and yet network consensus on the blockchain. What are your thoughts on the evolution of privacy?

John: We are the first to admit that health care is just getting started and things are not perfect at this early stage. A good example of the immaturity is evident in the conversations we have around public vs. private blockchains. The key distinction is that whereas permissioned blockchains are merely *distributed* solutions, open blockchains offer real *decentralization*. From a technical perspective, we choose the protocol based on the problem we are solving. We are comfortable with the concept of the blockchain actually representing a spectrum of trust-based transactional systems ranging from open and decentralized to private and distributed. We believe that the industry will move toward open and public blockchains, but it might take time to get there. Health care enterprises, much like financial services and other enterprises, value greater levels of control and, perhaps most important, levels of confidentiality that currently are not possible on open, truly decentralized blockchains. It is indisputable that for a range of business use cases, permissioned blockchains are an ideal fit. We believe it will take time for companies to become comfortable with the more traditional blockchain models, even though it's clear those models offer the greatest security.

It is important to critically examine where this preference for control can become a damaging and self-defeating prejudice. Open and decentralized blockchains are continuing to explode onto the scene. The top two cryptocurrencies alone account for $65 billion in market capitalization. Further, the newer trend of initial coin offerings (ICOs) have attracted over $1 billion via crowdfunding for mostly open source, decentralized blockchain platforms. The overwhelming level of interest in and financial backing for these open platforms cannot be ignored, nor should they be. They signal a massive opportunity for both individuals and businesses to think differently about the concepts of ownership, control of the network, and funding for infrastructure that supports the common good. This could be very important in funding platforms that challenge the current corporatocracy. In health care, we could fund massive public infrastructure projects that put the power of health and wellness in the hands of the true customers. That's powerful and it's hard to imagine how that would get funded through traditional means.

Setting aside technical issues of confidential transactions and PHI on blockchains, the prospect of health care businesses operating on open networks is truly daunting to many businesses. Centralization, consolidation, and ever-widening control of health care networks has been the dominant business strategy for the industry for decades, especially since the introduction of the Affordable Care Act. Business success in health care has primarily come from ever greater control of value chains focusing on covered lives, pharmaceuticals, claims, specialty networks, outpatient facilities, and supplies. It is clear that too much value is being extracted by value chain participants today. An open blockchain solution exposes these relationships and forces reintermediation with lighter, more nimble value-adding actors. The uncomfortable question that blockchain technology poses for the health care industry is the unconventional yet tangible opportunities that open, decentralized networks offer for true value-added health care services.

Vikram: As the blockchain matures, where do you see the main value being created? We spoke earlier about the fat protocols creating and capturing value at the architecture level, but what about the network?

John: In today's health care industry, owning and operating the network is the ultimate business goal. The consumers of health care services have very little market power with choices limited by arcane and opaque contractual relationships that define today's health care networks. They cannot assemble their own networks of providers and services; they cannot negotiate on price or other value-added services. Instead consumers are "steered." But an open, decentralized market for health care services would enable the consumer be free to make rational economic decisions. An open network has to be free of the perverse incentives that constrain choice and drive patient steerage. The essential difference between the status quo and the promise of decentralized networks is this: Running a decentralized network is not a business in and of itself.

So far health care has resisted the platform movement. In a short period of time we have watched as Uber, Airbnb, and others have disrupted several traditional industries. Health care leaders have watched those markets change with some comfort in the reality that their existing health care value chains are too complex, too regulated, and too sensitive to failure. Resting on these assumptions might be a mistake. The costs are becoming unsustainable and consumers are demanding another door.

Open blockchain platforms are a new reality that health care needs to recognize and embrace. Hashed Health plans to move in this direction and will continue to promote open solutions. These systems, protocols, and tools are maturing rapidly and will not go away. They are proving their economic viability. The platform itself is built on open source software. Organizations such as nonprofit foundations now have the means to raise sufficient funds to launch these platforms. Incentive and fee structures can be implemented to fund ongoing operations, making the platform truly self-sufficient. The core blockchain innovation of decentralized networks supporting common transactional systems can keep the platform in the centralized control of any single entity. Open governance models continue to be refined.

Value on open networks is defined purely by economic fundamentals of the services being offered. By contrast, the closed ecosystems of the health care industry seem to thrive on distorting true value by constraining choices. Tremendous cost and expensive administrative inefficiencies are in some sense a necessity designed to amass control over the network itself. By giving up control of the basic platform itself, health care enterprises can be economically rewarded by providing valued services with much lower burden of overhead and administrative cost.

The most important point is that open decentralized networks are not fundamentally incompatible with health care, despite privacy and regulatory concerns. Technical barriers will soon give way to innovations such as "zero-knowledge proofs" and other means of executing confidential blockchain transactions. The true barrier is an entrenched business mind-set that will become obsolete in time. It is not a matter of if, but when open health care networks will take root. In the short term, we need private networks to demonstrate value and move the conversation forward. Hashed will be a leader in developing these networks as a step-wise approach to the eventual reality that open blockchains will deliver the most disruptive and effective solutions. It is those enterprises that can give up white-knuckled control of the networks that will reap the greatest opportunity.

Vikram: Finally, I have to ask you about the recent ICO craze. Every blockchain company is trying to do an ICO, which reminds me of the late 1990s environment. Is Hashed Health going to do an ICO? What and how will the tokens be used?

John: We have also developed expertise in tokenization, and we believe that one or more of our products will have a meaningful, token-centric architecture. The concept of a token offering is exciting because it has the potential to fund infrastructure concepts for public health. Designed effectively, tokens can also help deliver on the promise of intelligent value exchange in health care. Programmable payments are an incredible opportunity to improve how money flows in health care today, creating better incentive structures that result in the alignment we have been missing for so long, especially when it comes to physician and patient behavior. Dr. Rasu Shrestha said it well: "At the end of the day, innovation is really about behavior change, whether it's a clinician putting in an order, that radiologist making a specific diagnosis or call on a finding,

or the patient making a decision to eat that muffin versus going for that salad. Innovation is about behavior change" (see https://www.healthcare-informatics.com/article/upmc-s-rasu-shrestha-innovation-about-behavior-change-technology-should-be-invisible).

We are in no rush to jump into the ICO craze, preferring to take our time and make sure the token mechanism is integral to the product and the token sale is offered in a way that satisfies the interests of our constituents while getting the token into the hands of those who will use it. No one can ignore the power of this innovation. A team of engineers anywhere in the world can make available a secure financial system with clear benefits over what exists today. It's a model that makes it difficult for traditional systems and conventionally funded startups to compete.

We feel confident that several of our use cases and pending partnerships have tokenization opportunities. We see tokens as a new, better container than what we have today. We are extremely interested in and excited by the emerging relationship between companies and tokens in the health industry. We are also interested to see if we can apply our same collaborative principles to the token distribution concept, especially in the areas of governance and coordination models.

The value that Hashed Health delivers is simple and necessary at this stage of market and product building. We are a health care company first, which means we have the patient as our primary consideration. Our team averages 15 years in health care technology. We know health care and we are able to connect challenges in health care with the possibilities of blockchain. This means we are really good at health care blockchain use cases. Second, in a world where the technology is changing rapidly, we do not lock customers into a specific technology. Not every protocol or middleware solution is perfect for a specific business problem. At Hashed, the protocol supports the problem. Third, our collaborative approach lowers the risk and increases the likelihood of success for projects. The "build it and they will come" model results in an expensive academic exercise. There's a better path where you can distribute the costs and the rewards across a network of collaborators who are organized around success. We have created a company and a business model that fits the spirit of the decentralized health solutions we build. Together we will innovate and together we will accelerate the meaningful productive use and realize the potential of blockchain in health care.

APPENDIX B

References

This appendix includes detailed references used to prepare each chapter.

Chapter 1

1. Nakamoto, Satoshi. "Bitcoin: A peer-to-peer electronic cash system." (2008): 28.
2. Nakamoto, Satoshi. "Re: Bitcoin P2P e-cash paper." The Cryptography Mailing List (2008).
3. Velde, François. "Bitcoin: A primer." Chicago Fed Letter Dec (2013).

Chapter 2

1. Böhme, Rainer, Nicolas Christin, Benjamin Edelman, and Tyler Moore. "Bitcoin: Economics, technology, and governance." The Journal of Economic Perspectives 29, no. 2 (2015): 213-238.
2. Bulkin, Aleksandr. "Explaining blockchain—how proof of work enables trustless consensus." Keeping Stock. May 3, 2016. https://keepingstock.net/explaining-blockchain-how-proof-of-work-enables-trustless-consensus-2abed27f0845.
3. Kroll, Joshua A., Ian C. Davey, and Edward W. Felten. "The economics of Bitcoin mining, or Bitcoin in the presence of adversaries." In Proceedings of WEIS, vol. 2013. 2013.
4. Nielsen, Michael. "How the Bitcoin protocol actually works." Data-driven Intelligence. December 6, 2003. http://www.michaelnielsen.org/ddi/how-the-bitcoin-protocol-actually-works/.
5. O'Dwyer, Karl J., and David Malone. "Bitcoin mining and its energy footprint." (2014): 280-285.

Chapter 3

1. "Bitcion Developer Reference." https://bitcoin.org/en/developer-guide.
2. Becker, Georg. "Merkle signature schemes, merkle trees and their cryptanalysis." Ruhr-University Bochum, Tech. Rep. (2008).
3. Franco, Pedro. Understanding Bitcoin: Cryptography, engineering and economics. John Wiley & Sons, 2014.

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7

Chapter 4

1. Buterin, Vitalik. "Ethereum: A next-generation smart contract and decentralized application platform." URL: https://github.com/ethereum/wiki/wiki/%5BEnglish%5D-White-Paper (2014).
2. Delmolino, Kevin, Mitchell Arnett, Ahmed Kosba, Andrew Miller, and Elaine Shi. "A programmer's guide to ethereum and serpent." URL: https://mc2-umd.github.io/ethereumlab/docs/serpent_tutorial.pdf (2015).
3. Ethereum Community. "Ethereum Homestead Documentation." Readthedocs. March 1, 2017. https://media.readthedocs.org/pdf/ethereum-homestead/latest/ethereum-homestead.pdf.
4. Wood, Gavin. "Ethereum: A secure decentralised generalised transaction ledger." Ethereum Project Yellow Paper 151 (2014).

Chapter 5

1. Atzori, Marcella. "Blockchain technology and decentralized governance: Is the state still necessary?." (2015).
2. Cuende, Luis, and Jorge Izquierdo. "Aragon Network: A Decentralied Infrastructure For Value Exchange." GitHub. April 20, 2017. https://github.com/aragon/whitepaper/blob/master/Aragon%20Whitepaper.pdf.
3. Merkle, R., 2015. DAOs, Democracy and Governance.

Chapter 7

1. Bonomi, Flavio, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. "Fog computing and its role in the internet of things." In Proceedings of the first edition of the MCC workshop on Mobile cloud computing, pp. 13-16. ACM, 2012.
2. Bylica, Paweł, L. Glen, Piotr Janiuk, A. Skrzypcaz, and A. Zawlocki. "A Probabilistic Nanopayment Scheme for Golem." (2015).
3. Dannen, Chris. "Smart Contracts and Tokens." In Introducing Ethereum and Solidity, pp. 89-110. Apress, 2017.
4. IEx.ec Team. "Blueprint For a Blockchain-based Fully Distributed Cloud Infrastructure." iEx.ec project. March 18, 2017. https://iex.ec/app/uploads/2017/04/iExec-WPv2.0-English.pdf.
5. Merriam, Piper. "Ethereum Computation Market 0.1.0 documentation." 2016. http://docs.ethereum-computation-market.com/en/latest/.
6. SOMN Team. "Supercomputer organized by network mining." SONM. March 19, 2017. https://sonm.io/SONM_TECHNICAL_WP.pdf.
7. Teutsch, Jason, and Christian Reitwießner. "A scalable verification solution for blockchains." (2017).

Chapter 8

1. Aarts, A. A., J. E. Anderson, C. J. Anderson, P. R. Attridge, A. Attwood, and Anna Fedor. "Estimating the reproducibility of psychological science." Science 349, no. 6251 (2015): 1-8.
2. Baker, Monya. "1,500 scientists lift the lid on reproducibility." Nature News 533, no. 7604 (2016): 452.
3. Begley, C. Glenn, and John PA Ioannidis. "Reproducibility in science." Circulation research 116, no. 1 (2015): 116-126.
4. Dreber, Anna, Thomas Pfeiffer, Johan Almenberg, Siri Isaksson, Brad Wilson, Yiling Chen, Brian A. Nosek, and Magnus Johannesson. "Using prediction markets to estimate the reproducibility of scientific research." Proceedings of the National Academy of Sciences 112, no. 50 (2015): 15343-15347.
5. Etz, Alexander, and Joachim Vandekerckhove. "A Bayesian perspective on the reproducibility project: Psychology." PLoS One 11, no. 2 (2016): e0149794.
6. Gezelter, J. Daniel. "Open source and open data should be standard practices." (2015): 1168-1169.
7. Open Science Collaboration. "Estimating the reproducibility of psychological science." Science 349, no. 6251 (2015): aac4716.
8. Pashler, Harold, and Eric–Jan Wagenmakers. "Editors' introduction to the special section on replicability in psychological science: A crisis of confidence?." Perspectives on Psychological Science 7, no. 6 (2012): 528-530.
9. Scannell, Jack W., and Jim Bosley. "When quality beats quantity: decision theory, drug discovery, and the reproducibility crisis." PloS one 11, no. 2 (2016): e0147215.

Chapter 9

1. Dubovitskaya, Alevtina, Zhigang Xu, Samuel Ryu, Michael Schumacher, and Fusheng Wang. "How Blockchain Could Empower eHealth: An Application for Radiation Oncology." In VLDB Workshop on Data Management and Analytics for Medicine and Healthcare, pp. 3-6. Springer, Cham, 2017.
2. Emily Vaughn. "A Universal Library for Health Care: Health Data Meets Blockchain Technology." Gem HQ Blog. June 20, 2016. https://blog.gem.co/blockchain-health-data-library-e53f930dbe93.
3. Ekblaw, Ariel, Asaph Azaria, John D. Halamka, and Andrew Lippman. "A Case Study for Blockchain in Healthcare: "MedRec" prototype for electronic health records and medical research data." In Proceedings of IEEE Open & Big Data Conference. 2016.
4. Mettler, Matthias. "Blockchain technology in healthcare: The revolution starts here." In e-Health Networking, Applications and Services (Healthcom), 2016 IEEE 18th International Conference on, pp. 1-3. IEEE, 2016.
5. Kuo, T. T., C. N. Hsu, and L. Ohno-Machado. "ModelChain: Decentralized Privacy-Preserving Healthcare Predictive Modeling Framework on Private Blockchain Networks." In ONC/NIST Blockchain in Healthcare and Research Workshop, pp. 26-7. 2016.

6. Yue, Xiao, Huiju Wang, Dawei Jin, Mingqiang Li, and Wei Jiang. "Healthcare data gateways: found healthcare intelligence on blockchain with novel privacy risk control." Journal of medical systems 40, no. 10 (2016): 218.

Chapter 10

1. Cachin, Christian. "Architecture of the Hyperledger blockchain fabric." In Workshop on Distributed Cryptocurrencies and Consensus Ledgers. 2016.

2. Chen, Lin, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. "On Security Analysis of Proof-of-Elapsed-Time (PoET)." In International Symposium on Stabilization, Safety, and Security of Distributed Systems, pp. 282-297. Springer, Cham, 2017.

3. Manuel Garcia. "Introduction to Blockchain and the Hyperledger Project." SlideShare. May 6, 2016. https://www.slideshare.net/ManuelGarcia122/introduction-to-blockchain-and-the-hyperledger-project.

4. Morgen Peck. "Do You Need a Blockchain?" IEEE Spectrum. September 29, 207. https://spectrum.ieee.org/computing/networks/do-you-need-a-blockchain.

5. Prisco, Giulio. "Intel develops 'Sawtooth Lake' distributed ledger technology for the Hyperledger project." Bitcoin Magazine (2016).

6. Sankar, Lakshmi Siva, M. Sindhu, and M. Sethumadhavan. "Survey of consensus protocols on blockchain applications." In Advanced Computing and Communication Systems (ICACCS), 2017 4th International Conference on, pp. 1-5. IEEE, 2017.

7. Sebastien Meunier. "When do you need blockchain? Decision models." Medium. August 4, 2016. https://medium.com/@sbmeunier/when-do-you-need-blockchain-decision-models-a5c40e7c9ba1.

8. Tracy Kuhrt. "Oscon 2017: Contributing to Hyperledger." SlideShare. May 12, 2017. https://www.slideshare.net/tkuhrt/oscon-2017-contributing-to-hyperledger.

9. Underwood, Sarah. "Blockchain beyond bitcoin." Communications of the ACM 59, no. 11 (2016): 15-17.

10. Vukolić, Marko. "Rethinking Permissioned Blockchains." In Proceedings of the ACM Workshop on Blockchain, Cryptocurrencies and Contracts, pp. 3-7. ACM, 2017.

11. Wüst, Karl, and Arthur Gervais. "Do you need a Blockchain?." IACR Cryptology ePrint Archive 2017 (2017): 375.

Chapter 11

1. Bob Summerwill, and Shahan Khatchadourian. "Enterprise Ethereum Alliance Technical Roadmap." Ethereum Enterprise Alliance. February 28, 2017. https://bobsummerwill.files.wordpress.com/2017/02/enterprise-ethereum-technical-roadmap-slides-final.pdf.

2. Chain Team. "Chain Protocol Whitepaper." Chain Developer Documentation. 2017. https://chain.com/docs/1.2/protocol/papers/whitepaper.

3. Chain Team. “The Ivy Language.” Chain Developer Documentation. 2017. https://chain.com/docs/1.2/ivy-playground/docs.

4. Daniel Larimer. “EOS.IO Technical White Paper.” GitHub. June 26, 2017. https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.

5. David Voell. “Quorum Architecture.” GitHub. October 16, 2017. https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum_Architecture_20171016.pdf.

6. David Voell. “Quorum Whitepaper.” GitHub. November 22, 2016. https://github.com/jpmorganchase/quorum-docs/blob/master/Quorum%20Whitepaper%20v0.1.pdf.

7. David Voell. “Quorum Blockchain: Presentatino to Hyperledger Project.” GitHub. November 21, 2016. https://github.com/jpmorganchase/quorum-docs/blob/master/Blockchain_QuorumHyperledger_20160922.pdf.

8. Ian Grigg. “EOS - An Introduction.” EOS. July 5, 2017. https://eos.io/documents/EOS_An_Introduction.pdf.

Index

A

Altcoins, 191
Ambrosus, 194
Ark Invest, 131–132

B

Birch–Brown–Parulava model, 144, 146
Bitcoin, 2, 68
Bitcoin community, 186
Bitcoin exchange, *see* Digital currency exchanges (DCEs)
Blockchain industry
  computational logic, 185
  distributed database, 184
  irreversibility of records, 185
  peer-to-peer transmission, 184
  SCI, 184
  transparency with pseudonymity, 184
Blockchain(s)
  community, 67
  database, 145
  disintermediation, 145
  forks, 23–24
  no trust, 145
  SPV (*see* Simple payment verification (SPV))
  transaction interdependence, 145
  transactionworkflow (*see* Transactions)
  writers, 145
Blockchain technology
  Accenture, 194
  clinical trials
    asset metadata, 116
    benefits, 119
    coloring scheme, 116
    comparing drug efficacies, 115
    integration, 118
    postprocessing, 115
    reputation system, 117
    researcher registering, 116
    rule engine, 116
    scarcity, 116
    server–client interaction, 117
    trail registration, 115
  pharmaceutical drug tracking, 122–123
  reproducibility crisis
    cancer biology, 114
    DDI, 114
    minimum publishing standards, 114
    overview of positive and negative data, 112–113
  reputation system
    colored coin protocol, 122
    evaluator function, 120
    postprocessing unit, 122
    rep_score, 119–120
    third-party services, 119
Brass Golem, 79
Broker contract, 85–86
Business Network Archive (BNA), 148–149
Buterin's concept, 68
Buyer–hub–miner interactions
  hubs pool, 102
  hub wallet, 103
  overview, 103
  verification, 102

C

Capital One, waste management, 131–132
Cello, 140
Chain core
  characteristic, 160
  Click Join Network, 162
  configuring, 162
  confirmation of asset, 173
  contracts, 160
  difference between Ethereum and, 172
  HSM keys, 163
  Ivy, 161
  key pair, 172

© Vikram Dhillon, David Metcalf, and Max Hooper 2017
V. Dhillon et al., *Blockchain Enabled Applications*, https://doi.org/10.1007/978-1-4842-3081-7

Chain core (*cont.*)
navigation menu, 162
select asset, 173
Spend From Account
Ivy, 172
start trade, 170
transaction summary, 171
transferring ownership, 171
traditional services, 161
transactions
account summary, 167
asset creation, 165
assign asset to account, 169
create asset, 168
create new account, 167
default assets, 164
MockHSM keys, 163–164
multikey authentication, 165–166
new screen, 168
verification, 169
unlocking asset, 174
Coin Telegraph, 186
Constellation, 176
Counterparty, 1
Counterparty risk, 1
Crowdfunding smaller investment, 195
Cryptocurrencies, 26, 189
Cryptocurrency community, 191
Crypto-equity, 194
Cypherpunk community
Bitcoin protocol, 3
block, 4
Hashcash, 3
MVC, 4–5

D

Data Discovery Index (DDI), 114
Decentralized application (DApp), 47
Decentralized autonomous organization (DAO), 44, 47, 68, 99
Aragon kernel, 48
automated entities, 69
child, 72
DAO/company walkthrough
fundraising and bylaws, 63–66
issuing shares, 54–63
setting up, 50–54
debate, 75–76
hack
iterative process, 75
Payout(), 74
prevention, 72
Solidity contract, 73
split feature, 73
vulnerability, 72
to withdraw funds, 74–75
withdrawRewardFor(), 73–74
ICO for, 72
identity management, 49
Slock.it blog, 70–71
Solidity contract, 71
split, 76–77
team, 69–70
uint proposalDeposit, 71
vote function, 71
Delegated proof-of-stake (DPoS)
approval voting, 155
block producers, 155
block production, 155
consensus algorithms, 156
goal, 155
network support, 155
producing blocks, 156
roles, 155
stakeholders, 155
vs. traditional PoS algorithms, 154
Desktop grid, 106
Digital currency exchanges (DCEs), 189
Disintermediation, 145
Double spending, 2
Drug Supply Chain Security Act (DSCSA), 123

E

E-Fast, 107, 108
EOS blockchain, 151, 152
DPoS, 154–156
free access, 152
message handling scripts, 152
parallel execution
advantage, 159
block cycle, 157–158
block producer, 157
latency, 157
permission mapping
message module, 154
ownership portion, 154
scheduling
block producer, 160
subjective measurement, 160
virtual machine, 159–160
social media, 153
Steem, 153
updates and forks, 152
user base, 152
ERC20 standard, 80

Ethereum
accounts, 27, 29–30
alternate currencies, 44
Bitcoin, 26–27
Chain core and, 172
cryptocurrencies, 26
DApps
Geth and Mist, 44
IPFS-like system, 43
stack and interfaces, 42–43
structure, 42
EVM (*see* Ethereum Virtual Machine (EVM))
hardware-based mining, 26
mid-2013, 25
online forums, 25
scripting language, 26
state, storage and gas, 30–33
Ethereum Computational Market (ECM)
defined, 83
Fibonacci sequence, 88
finalized status, 83
firm *vs.* soft resolution, 83
gas charges and reward system, 87
life cycle, 84
marketplace
address executable, 85
address requester, 85
answerRequest function, 86
broker contract, 85–86
bytes32 resultHash, 85
execution contract, 85
factory contract, 85–86
initializeDispute function, 87
overview of market, 86
uint payment, 85
uint requiredDeposit, 85
uint softResolutionBlocks, 85
uint status, 85
on-chain, 83
pending status, 83
resolutions, 83
Ethereum Computer, 69
Ethereum Enterprise Alliance (EEA), 151
enterprise protocol, 180–181
Quorum, 175–176, 180–181
Ethereum tokens
access, 80
ICO, 81
overview of blockchain stack, 81–82
private key, 81
Ethereum Virtual Machine (EVM)
blockchain-based service, 41–42
concept, 33
contract method, 34
design, 33
deterministic nature of, 34
different nodes, 35
execution
arbitrary code, 33
contract code, 34
full node, 34
Solidity programming language, 36–38
world computer model, 38–40

F

Fabric
defined, 139
features, 143
Factory contract, 85
Fat protocols, 79
Fibonacci sequence, 88
FinTech, 189
Fog computing, 96
Fog computing cloud, 99

G

Gem, 131–132
Global financial system, 183
Goldman Sachs, 194
Golem network
application registry
DApps, 91
traditional validators, 91
vanguard validators, 91
whitelists/blacklists, 91
decentralized farm, 89
technology stack, 89–90
transaction framework
GNT, 91, 94
lottery system, 92–93
microservices, 92
nanopayments, 91
TrueBit's verification game, 94–95
types, 90
Golem Network Token (GNT), 91, 94
Graphics processing unit (GPU), 97

H

Hardware security module (HSM), 161
Hashcash, 3
Health care
patient visit workflow (*see* Patient visit workflow)
payer–providers–patient model (*see* Payer–provider–patient model)

High-performance computing (HPC), 79
Homechain company, 194
Howey Test, 190
Hyperledger Composer
  assets, 148
  blockchain-based prototype, 147
  BNA, 148–149
  concepts, 148
  participants, 148
  transaction processors, 148, 149
Hyperledger Project, Linux Foundation, 139

I

IBM model, 144, 146
iEx.ec
  E-Fast, 107, 108
  efficiency, 106
  matchmaking algorithm, 107
  off-chain, 106
  overview, 108
  pluggable nodes, 106
  resilience, 106
  scheduler algorithm, 107
  sidechain, 107
  SLA, 106
  XtremWeb-HEP, 106
Immutable state model, 172
Indy, 140
Initial coin offerings (ICOs), 68
  bitcoin community, 186
  Coin Desk, 186
  Coin Telegraph, 186
  cryptocurrency community, 191
  Ethereum tokens, 81
  Founders: Entrepreneurs, 190–191
  IPO, 185
  practices for, 190
  pros, 190–191
  risks of investing, 192
  SEC, 189
  SONM, 186–188
  venture capitalist and, 185
Initial public offering (IPO), 185
Internet of Everything (IoE), 96
Internet of Things (IoT), 96
Investopedia, 192
Iroha, 139
Ivy, 161, 172

J

JPMorgan, 175, 177

K

Keybase Filesytem (KBFS), 49
KICKCO, 195
Kracken web site, 189

L

Linux Foundation, 139, 141
Lottery payment system, 92–93

M

Matchmaking algorithm, 107
Mining
  ASICs, 13
  block-header, 8–9
  hardware, 12–13
  PoW problem, 9–11
  process, 7–8
  roles, 7
Model-View-Controller (MVC), 4

N, O

Nongovernmental organizations (NGOs), 47
Numerai, 194

P

Patient visit workflow
  hash, 129
  hot switching, 131
  initial, 128
  SOAP note, 129
  specialist integration, 130, 133
  tests, 130
Payer–provider–patient model
  defined, 125
  insurance company, 127
  overview, 126
  scenarios, 126
Peer-to-peer transmission, 184
Permission mapping, 153–154
Pharmaceutical drugs, tracking, 122–123
Pi-calculus, 99
Plug-ins circuit board, 97
Premine, 81
Proof-of-concept (PoC), 193
Proof-of-stake (PoS), 154

Q

Quorum
area of focus, 181
connection layer, 176
Constellation, 176
node permission, 175
overview and key innovations, 178
performance, 175
pluggable consensus, 176
privacy, 175
privateFor, 176
private transactions, 176, 179
QuorumChain, 175–176
raft-based consensus, 175–176

R

Raft-based consensus, 175–176
RegApp, 100, 106
Regulation technology (RegTech)
benefits, 192
Deloitte, 193
EY's publication, 192
Investopedia, 192
PoC, 193
Reproducibility, 111

S

SALT lending platform, 194
Sawtooth, 139
Scheduler algorithm, 107
Securities and Exchange Commission (SEC), 185
Securities Law Framework for Blockchain Tokens, 190
Service-level agreement (SLA), 106
Service protocol, 104
Shared data layer, 80
Simple payment verification (SPV), 21–23
Smart contract system
crowdfunding efforts, 100
DAO, 99
hub factory, 100
hub wallet, 100
overview, 101
PayOut app, 100
pi-calculus, 99
RegApp, 100
SONM token, 99
whitelist, 100
Smith & Crown Index (SCI), 184
Steem, 153
Supercomputing Organized by Network Mining (SONM), 79
BIOS, 97
buyer–hub–miner interactions, 101–103
connected peripherals, 97
CPU/processor, 97
defined, 96, 186
fog computing, 96
fog computing cloud, 99
GPU, 97
hard disk, 97
ICO execution steps, 186–188
IoE, 96
IoT, 96
load balancer, 99
plug-ins board, 97
serial bus, 97
smart contract system
crowdfunding efforts, 100
DAO, 99
hub factory, 100
hub wallet, 100
overview, 101
PayOut app, 100
pi-calculus, 99
RegApp, 100
token, 99
whitelist, 100
SOSNA, 104, 106
world computer in, 98
Yandex.Cocaine, 97
Superglobal Operation System (SOSNA)
components, 104
defined, 104
grid architecture, 104
locator, 104
overview, 105
service, 104
Swarm company, 194
Sybil attack, 86
System-on-a-chip (SoC) systems, 70

T

Token sales, *see* Initial coin offerings (ICOs)
Transactions
block header, 16–17
consensus, 16
inputs and outputs, 18
internal consistency, 15
mobile wallets, 21
UTXO, 18–21
TrueBit's verification game, 94–95

U

Universal Sharing Network (USN), 69
Unspent transaction output (UTXO), 18–21

V

Vanguard validators, 91
Venture capitalist, 185

W, X

Waste management, 132
Whitelist mechanisms, 89, 91, 100
Wüst–Gervais model, 144, 147

Y

Yandex.Cocaine, 97

Z

Zero-knowledge proofs (zk-SNARKs), 177
Zero Knowledge Succinct Noninteractive Arguments of Knowledge (zk-SNARKs), 177